# FROM DESIGN TO LEARNING: EVOLVING RESEARCH WRITING HARNESSES FROM EXPERT PAPERS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Turning unstructured research materials into coherent manuscripts is an important part of AI-assisted research. Existing writing systems rely on predefined agentic workflows and skills shaped by a small number of designers, leaving open whether better writing practices can be discovered through broader writing experience. We introduce EvoWriter, a meta-harness that enables research writing systems to learn from the writing experience embodied in expert papers. EvoWriter operates around a writing harness that manages manuscript planning, drafting, and revision, and iteratively improves its agentic workflows and skills through writing practice. Specifically, it compares generated manuscripts with expert papers to identify potential improvements, translates these findings into reusable workflow and skill updates, and retains updates that improve subsequent writing. We study this evolution process using 100 research writing tasks drawn from distinguished papers across five major AI conferences and eight contribution types, with separate subsets for evolution, update selection, and final evaluation. Through this process, EvoWriter evolves the initial writing harness into WritingHarness, an executable system for generating complete manuscripts from new research materials. Across automatic and independent human evaluations, the WritingHarness outperforms baseline writing systems, producing clearer scientific arguments and better-organized evidence while preserving the supplied results and claim boundaries. These results show that writing experience distributed across expert papers can guide the continued improvement of research writing systems beyond predefined human designs.

## 1 INTRODUCTION

Large language models increasingly help generate research ideas and run experiments, making the research process faster (Ghareeb et al., 2026). In AI research, The AI Scientist combines idea generation, coding, and experimentation in an automated pipeline (Lu et al., 2026), while autoresearch lets an agent search for model improvements through repeated experimentation (Karpathy, 2026). As more research activities become automated, the resulting notes, experimental records, figures, and findings also need to be organized into papers that readers can understand and assess. The use of generative AI for academic writing is becoming increasingly common (Liang et al., 2025), helping researchers improve expression and reduce the effort of manuscript preparation (Li et al., 2024; Hu et al., 2025). In practice, such assistance often focuses on relatively local writing tasks, such as drafting, language polishing, and revision (Jourdan et al., 2023; Liang et al., 2024; Chamoun et al., 2024). However, turning scattered research materials into a complete manuscript requires more than assembling them into a paper: a system must organize an argument around the research contribution, maintain coherence across sections, and make the relationship between claims and evidence clear (Song et al., 2026; Qian et al., 2026a).

To address this challenge, AI-assisted research writing has expanded from local drafting, polishing, and revision (Jourdan et al., 2023; Liang et al., 2024; Chamoun et al., 2024) to coordinated agentic workflows for generating complete manuscripts (Weng et al., 2025a; Qian et al., 2026a). For example, PaperOrchestra (Song et al., 2026) proposed by Google uses a multi-agent workflow to turn research materials into a complete manuscript with a literature review, figures, and tables . Writing knowledge can also be encoded as reusable skills that guide specific writing tasks (Orchestra Research, 2026a). Together, such agentic workflows and skills form the basis of a writing harness that organizes



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.

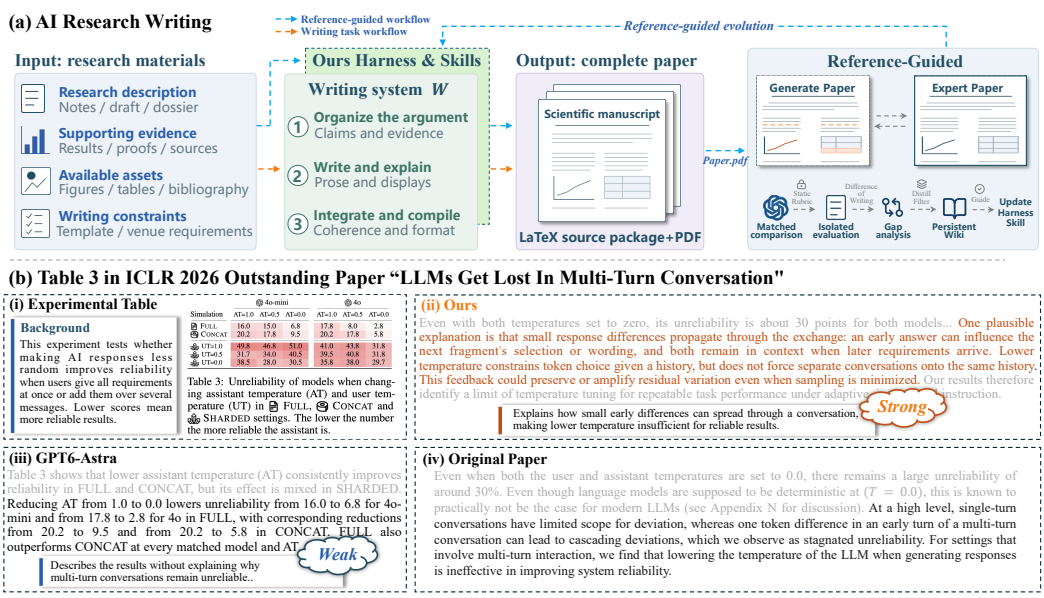



Figure 1: (a) Research writing task and reference-guided evolution. (b) A table-analysis example on multi-turn reliability, showing how the same source table is analyzed by different writers: (i) source table, (ii) ours, (iii) GPT-6 Astra, and (iv) the original paper.

the manuscript generation process. However, existing writing systems rely on predefined agentic workflows and skills shaped by a small number of designers, leaving open whether better writing practices can be discovered through broader writing experience. Scientific writing involves many choices in argument, evidence, and writing procedures that are difficult to determine in advance. Rather than fixing these choices in a human-designed harness, we ask whether writing systems can refine them from existing writing experience.

To this end, we introduce **EvoWriter**, a system that evolves research writing systems by learning from the writing experience embodied in expert papers as shown in Figure 1(a). Expert papers provide concrete examples of how researchers organize scientific arguments and present evidence across diverse research problems. EvoWriter learns from this writing experience to explore better practices and gradually incorporate validated improvements into a reusable writing system. To support this evolution, EvoWriter adopts a meta-harness architecture (Lee et al., 2026; Zhang et al., 2026; Lin et al., 2026; Luo et al., 2026), placing an optimization layer around an existing writing harness. The inner harness turns research materials into manuscripts through planning, writing, and revision. The outer meta-harness compares these generated manuscripts with expert papers to identify potential improvements in the underlying agentic workflows and skills, translates the findings into reusable updates, and validates whether they improve subsequent writing. Accepted updates are gradually incorporated into **WritingHarness**, an executable writing system that turns new research materials into complete manuscripts. In this way, EvoWriter uses expert papers to guide the exploration and accumulation of better writing practices beyond the initial human-designed harness.

We evaluate EvoWriter following PaperOrchestra's AI research writing task (Song et al., 2026) as shown in Figure 1(a), using 100 writing tasks spanning five major AI conferences and eight contribution types. Our experiments show that learning from expert-paper experience consistently improves the initial human-designed writing harness and generalizes to held-out writing tasks. The evolved WritingHarness produces manuscripts with clearer scientific arguments and better organization and use of evidence, while preserving the supplied experimental results and claim boundaries. Independent human evaluation also further confirms these improvements. Further analysis shows that expert-paper guidance and a separate evaluator lead to more effective evolution. We also find that evolving writing skills brings further gains after harness-level improvement plateaus. These results show that writing systems can improve beyond fixed human designs by learning from expert writing experience.

## 2 Related Work

**Research writing systems.** AI-assisted writing systems have progressed from supporting individual writing tasks to organizing complete research manuscripts. SurveyGen-I and LiRA focus on literature synthesis, using planning, memory, and specialized agents to organize information across multiple sources (Chen et al., 2025; Go et al., 2026). CycleResearcher incorporates iterative review into research and manuscript generation (Weng et al., 2025a), while PaperOrchestra uses a multi-agent workflow to produce complete papers from research materials, including literature reviews, figures, and tables (Song et al., 2026). Another line of work encodes writing knowledge as reusable skills that guide specific research and writing tasks (Orchestra Research, 2026a). EvoWriter builds on these approaches to manuscript generation, but focuses on whether the writing system itself can improve beyond its initial human design. We use expert papers as writing experience to explore better writing practices and identify workflow and skill updates that can improve the underlying writing harness. By validating and reusing these updates across manuscripts, EvoWriter turns experience from expert writing into reusable improvements to the writing system.

**Skill and harness evolution.** Recent work has begun to treat the reusable procedures of an agent as objects that can themselves be optimized. SkillOpt improves reusable skills through execution feedback, while WikiSkill extracts and refines procedural knowledge from experience for reuse across tasks (Yang et al., 2026; Tang et al., 2026). At the system level, meta-harness methods place an optimization process around an agent harness so that its agentic workflows and procedures can be revised through repeated execution and evaluation (Lee et al., 2026; Zhang et al., 2026; Lin et al., 2026). Most of these methods focus on tasks with relatively explicit execution or evaluation signals, while research writing depends strongly on human preferences that are difficult to specify in advance. EvoWriter investigates what experience can effectively guide evolution in such preference-driven tasks, showing how research-writing systems can actively learn and evolve from existing human writing experience.

## 3 Task and Dataset

### 3.1 Task Formulation

Following Song et al. (2026), we formulate research writing as mapping supplied research materials to a complete manuscript (Figure 1, upper panel). For paper $i$, the input $x_i = (d_i, e_i, a_i, c_i)$ contains a research description $d_i$, supporting evidence $e_i$, available assets $a_i$ (experimental results, and bibliography), and writing constraints $c_i$. The writing system $W$ produces LaTeX sources with supporting assets and a compiled PDF:

$$p_i = (p_i^{\mathrm{src}}, p_i^{\mathrm{pdf}}) = W(d_i, e_i, a_i, c_i). \tag{1}$$

The system may revise wording and organization, but must preserve supplied measurements, assumptions, and claim boundaries without inventing findings.

### 3.2 Dataset

We construct 100 paper-material packages from ICML, NeurIPS, ICLR, ACL, and CVPR. All papers received distinguished recognition at their respective venues, with at least an Oral or Spotlight designation. The collection covers eight common contribution types, including method, analysis, theory, benchmark, dataset, survey, application, and system papers. Each package pairs a research dossier with available supporting evidence and assets for the writing task in Section 3.1. Detailed dataset statistics and composition are provided in Appendix A.1.

We divide the collection into 40 learning evolution, 15 selection, and 45 evaluation tasks, balancing venue and contribution type while keeping each paper's materials together. The evolution set provides expert-paper experience for improving the writing harness, the selection set determines which updates are retained, and the evaluation set measures performance on unseen papers with the system fixed.

# 4 METHOD

We introduce **EvoWriter**, a framework that evolves a human-designed scientific-writing system by learning from expert papers. We first formulate how EvoWriter evolves the agentic workflows and skills of a writing harness through an outer optimization loop . We then describe how expert papers provide concrete writing experience for proposing and validating reusable updates, followed by our progressive evolution strategy that moves from broad harness exploration to focused skill refinement. Finally, we present the resulting **WritingHarness** and explain how it uses the evolved agentic workflows and skills to produce manuscripts from new research materials.

## 4.1 EVOLVING AGENTIC WORKFLOWS AND SKILLS

A writing harness coordinates planning, drafting, and revision. Its agentic workflows determine how these activities interact, while its skills provide reusable instructions for individual writing operations. At evolution round $t$, the harness $H_t$ and skill library $S_t$ produce a manuscript $p_{i,t} = H_t(M, S_t, x_i, r_i)$ from research materials $x_i$, with Writer model $M$ and generation seed $r_i$. EvoWriter places an outer optimization loop around this writing process. The inner loop revises the current manuscript; the outer loop changes the shared writing state $z_t = (S_t, H_t)$ used for subsequent manuscripts. Its objective is expected manuscript quality across writing tasks:

$$J(z) = \mathbb{E}_{x,r}[Q(H(M, S, x, r), x)], \qquad z^\star \in \underset{z=(S,H)\in\mathcal{Z}}{\arg\max}\ J(z). \tag{2}$$

Here $Q$ evaluates manuscript quality against the supplied materials under a fixed, reference-free protocol, and $\mathcal{Z}$ contains states allowed by the evidence and access contracts. The expectation covers writing tasks and generation seeds. The Writer model and each task's research evidence remain fixed within an evolution run.

## 4.2 LEARNING FROM EXPERT PAPERS

Expert papers make writing choices observable in context: how a research question is motivated, how evidence supports a claim, and how sections carry an argument forward. EvoWriter uses these examples to guide its search for writing practices. Each learning task pairs research materials with according expert manuscript. The inner harness writes from the materials, and an isolated evaluator compares the resulting manuscript with its expert counterpart under a fixed rubric. The resulting gap report $g_{i,t}$ identifies potential improvements in explanation, organization, and evidence presentation. Figure 2(a) shows how these comparisons lead to candidate updates and their validation.

**From comparison to reusable guidance.** Gap Analysis consolidates training reports into writing problems, proposed operations, and conditions under which those operations apply. For example, a recurring failure to explain a problem before introducing its formal definition can motivate a section-writing instruction that establishes the problem first. Reference-only facts and copyable wording are removed, and conflicting expert choices remain conditioned alternatives. A persistent Wiki records these lessons with their supporting evidence and uncertainty.

Permitted records supply reference-guided knowledge $K_t^{\mathrm{ref}}$:

$$K_t^{\mathrm{ref}} = \mathrm{GapAnalysis}(\{g_{i,t}\}_{i\in\mathcal{I}_{\mathrm{train}}}), \qquad \widetilde{z}_{t+1} = \mathrm{Update}(z_t, K_t^{\mathrm{ref}}; \mathcal{U}_t). \tag{3}$$

Here $\mathcal{I}_{\mathrm{train}}$ indexes training tasks and $\mathcal{U}_t$ limits editable components. Update Proposal translates the guidance into bounded changes to those components. The Writer receives only author materials, the fixed evidence interface, and approved active skills; expert papers and the Wiki remain outside its context.

**Validation and accumulation.** An expert-paper comparison suggests a writing practice; its value is assessed through subsequent writing. Update Validation generates complete manuscripts with the current and candidate states from the same held-out validation materials. Reference-free checks assess quality, fidelity, readiness, non-regression, and lower-tail performance. After these checks, an approved candidate becomes $H_{t+1}$; otherwise $H_{t+1} = H_t$. Proposal admission and promotion are distinct decisions, and a higher aggregate score alone is insufficient. Evaluation Archive retains

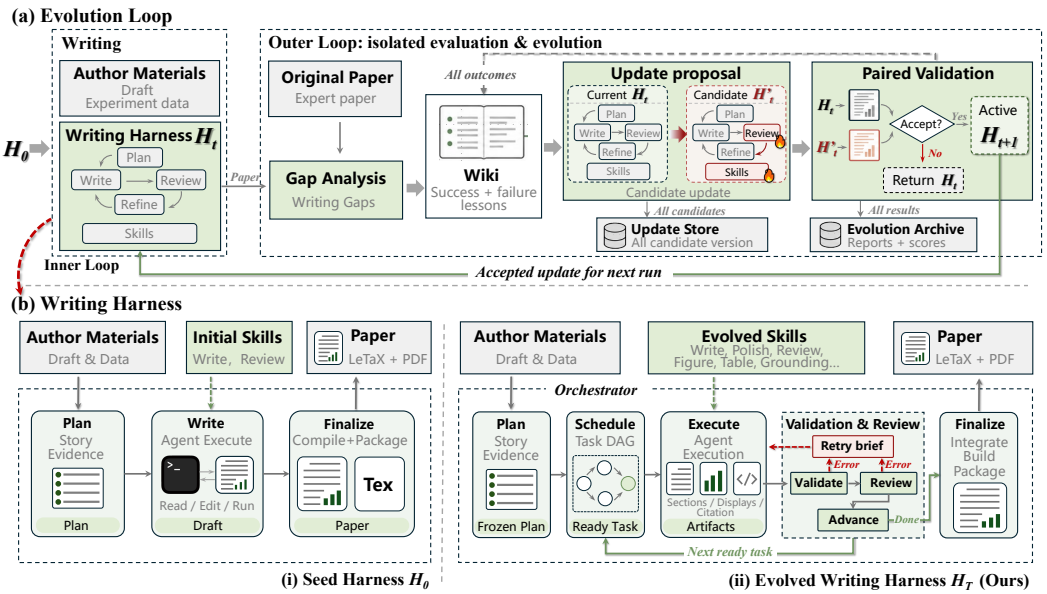



Figure 2: EvoWriter's method overview. (a) The isolated outer loop compares drafts with expert papers, proposes updates, and validates candidates before promotion. (b) The seed writing harness($H_0$) and the WritingHarness(ours) used to produce manuscripts from new author materials.

reports, and Update Store preserves all candidates, their parent versions, exact patches, and evidence lineage. Authorized summaries of accepted and rejected updates return to the Wiki to inform later proposals. Thus, expert examples guide exploration and validation governs what the system accumulates.

### 4.3 From Workflow Exploration to Skill Refinement

EvoWriter first evolves the harness, then refines skills under the retained workflow. As the harness grows and its modules mature, further broad updates may offer diminishing returns or disrupt effective behavior under a fixed budget. Freezing these modules focuses the same budget on improving their skills. Our evolution-schedule comparison (Section 5.2, Figure 4B) supports this choice: continued harness updates regress slightly, while switching to skills improves writing. Both stages use the same expert-paper guidance and validation loop.

**Harness evolution.** The initial stage can revise both the structure of the writing process and the instructions used within it. Its scope $\mathcal{U}_H$ includes skills, prompts, task scheduling, memory, and tools:

$$(\widetilde{S}_{t+1}, \widetilde{H}_{t+1}) = \mathrm{Update}(z_t, K_t^{\mathrm{ref}}; \mathcal{U}_H). \tag{4}$$

The base model, research evidence, rubric, and evaluator remain fixed. Each accepted harness change establishes a new version. At the transition to skill refinement, the retained state $z_m = (S_m, H_m)$ supplies the workflow on which further updates build.

**Skill evolution under a fixed harness.** Freezing the harness at $H_m$ preserves task scheduling and tool use while allowing further changes to writing instructions. The narrower scope $\mathcal{U}_S$ permits only skill edits:

$$(\widetilde{S}_{t+1}, H_m) = \mathrm{Update}((S_t, H_m), K_t^{\mathrm{ref}}; \mathcal{U}_S), \qquad \mathcal{U}_S \subseteq \mathcal{U}_H. \tag{5}$$

This stage focuses the search on reusable operations, such as explaining motivation or interpreting experimental evidence, within the retained workflow. Only approved active skills reach the Writer. Within each frozen harness epoch, the skill bundle is the sole mutable component; a harness change requires a new epoch.

### 4.4 WRITING WITH THE WRITINGHARNESS

The retained workflow and skill updates form the WritingHarness (Figure 2(b)). Compared with the linear seed harness, it introduces a persistent task graph with dependency-aware execution, validation, and revision. **Planning.** Instead of planning immediately before drafting, the evolved harness first produces a frozen plan that organizes claims and their supporting evidence, providing a stable basis for downstream tasks. **Scheduling and Execution.** While the seed harness writes largely in sequence, the evolved harness decomposes the plan into dependent tasks and schedules each when its prerequisites are ready. Evolved skills further support writing, grounding, figures, tables, and polishing. **Validation and Review.** Rather than reviewing only during writing, each task now passes a validation gate before the graph advances. Notably, evolution introduces *cold-reader review*: since the writer sees the full manuscript context, it may use concepts not yet explained to the reader. The cold reader therefore reviews each section using only the information available up to that point, and failed tasks are returned for revision. **Integration.** Finally, accepted artifacts are integrated into the complete manuscript, with claims, citations, terminology, and presentation reconciled across sections.

## 5 EXPERIMENTS

We separate writing outcomes from the process that updates the writing system. Writing Analysis compares complete systems and controls for coding host and base model. Evolution Analysis examines guidance, update schedule, and evaluator identity. A final human study section covers manuscript preference and its relation to Writing Overall.

**Experimental setup.** We split the 100-paper dataset into 40 development, 15 selection, and 45 test papers. The complete-system comparison and independent human evaluation are conducted on the 45-paper test set. For controlled evolution experiments, we use a separate 20-paper *Evolution Analysis Set*, split into 10/4/6 papers for evolution, update selection, and evaluation, respectively. Unless otherwise specified, GPT-5.5 powers the inner writing loop, while DeepSeek-V4-Flash powers the outer evolution loop.

Following PaperOrchestra (Song et al., 2026; Yamada et al., 2025) , we use **Clarity** and **Presentation** to assess general writing quality. We additionally evaluate **Experience** (ease of following and understanding the paper), **Evidence** (organization of claims and supporting evidence), **Retention** (recall of original main-text evidence in the generated main text), and **Support** (the proportion of claims supported by the provided materials). **Overall** equally averages all six normalized measures. **Accept.** reports the predicted acceptance rate and is excluded from both aggregate scores.

**Compared systems.** We compare against four groups of baselines: single-agent writing systems (OpenAI, 2026; Anthropic, 2026), frozen skill-assisted systems (Qian et al., 2026b; Orchestra Research, 2026b; Imbad0202, 2026; Yuan1z0825, 2026), multi-agent scientific-writing systems (Song et al., 2026; Yamada et al., 2025; Liu et al., 2026; Weng et al., 2025b), and experience-augmented baselines that use the evolution papers to prepare frozen writing guidance. All systems receive the same author materials and evidence interface, and skill-assisted and multi-agent baselines use GPT-5.5 unless otherwise specified.

### 5.1 WRITING ANALYSIS

**Complete-system performance.** Table 1 compares systems on the same test papers. Original (GT) is a reference in the native score units, not a score ceiling. Overall, stronger agents, writing skills, and multi-agent systems improve manuscript quality, but a clear gap to expert papers remains. Among all generated systems, our method achieves the strongest results across all six writing measures.

More importantly, the evolved harness substantially outperforms its human-predefined Seed Harness and existing multi-agent systems, with consistent gains in reading quality, evidence use, and retention. This suggests that our evolution process can discover writing procedures beyond the initial human design. Reference augmentation brings smaller improvements to existing systems, further showing the importance of turning expert-paper experience into reusable workflow and skill updates.

| Method | Reading | | | Capability | | | Summary | |
|---|---|---|---|---|---|---|---|---|
| | Clarity (1–4) | Presentation (1–4) | Quality (1–10) | Evidence (1–5) | Retention (%) | Support (%) | Overall (0–100) | Accept. (%) |
| **Reference Expert Paper** | | | | | | | | |
| Original (GT) | **3.75** | **3.78** | **8.79** | **4.50** | **100.00**† | **97.40** | **92.63** | **84.44** |
| **Single Agent** | | | | | | | | |
| Codex (GPT-5.6 Sol) | 2.70 | 2.74 | 5.65 | 2.40 | 51.36 | 78.32 | 55.17 | 28.89 |
| Codex (GPT-5.5) | 2.55 | 2.60 | 5.32 | 2.22 | 47.20 | 75.32 | 51.00 | 24.44 |
| Claude Code (Sonnet 5) | 2.82 | 2.86 | 5.95 | 2.62 | 57.80 | 81.92 | 59.65 | 37.78 |
| **Single Agent with Skills** | | | | | | | | |
| Spark-to-Paper Skills | 3.22 | 3.25 | 7.11 | 3.40 | 66.85 | 89.77 | 72.25 | 57.78 |
| ML Paper Writing | 3.00 | 3.02 | 6.55 | 3.10 | 60.75 | 86.31 | 65.87 | 51.11 |
| Academic Research Skills | 2.88 | 2.90 | 6.30 | 2.90 | 56.72 | 83.07 | 62.03 | 46.67 |
| Nature Skills | 3.07 | 3.10 | 6.69 | 3.20 | 60.83 | 87.17 | 67.54 | 51.11 |
| **Multi Agent System** | | | | | | | | |
| PaperOrchestra (Song et al., 2026) | 3.30 | 3.32 | 7.36 | 3.50 | 71.30 | 91.60 | 75.01 | 66.67 |
| AI Scientist-v2 (Yamada et al., 2025) | 3.15 | 3.18 | 6.91 | 3.15 | 65.57 | 88.27 | 69.60 | 53.33 |
| AutoResearchClaw (Liu et al., 2026) | 3.31 | 3.34 | 7.39 | 3.55 | 70.89 | 90.89 | 75.26 | 62.22 |
| DeepScientist (Weng et al., 2025b) | 3.28 | 3.30 | 7.27 | 3.60 | 70.07 | 90.66 | 74.68 | 62.22 |
| **Reference-Augmented System** | | | | | | | | |
| SeedHarnes (GPT-5.6) | 2.90 | 2.95 | 6.24 | 2.86 | 58.51 | 84.09 | 62.61 | 42.22 |
| ML Paper Writing | 3.04 | 3.06 | 6.68 | 3.13 | 63.91 | 86.92 | 67.31 | 51.11 |
| DeepScientist (Weng et al., 2025b) | 3.32 | 3.34 | 7.41 | 3.64 | 73.65 | 91.72 | 76.32 | 66.67 |
| **Ours** | **3.52** | **3.54** | **8.14** | **4.00** | **81.84** | **94.88** | **83.29** | **75.56** |

Table 1: Writing performance on the 45-paper test set (↑). Original (GT) denotes the expert-written reference paper, while bold values indicate the best generated results. Reference-augmented systems have access to the evolution papers during learning

**Controlled Writing Evaluation.** We evaluate the same frozen WritingHarness across OpenCode and Claude Code, using five Writers per agent on the 45-paper test split with matched inputs and generation limits. The Figure 3 shows consistent improvements across all tested agent–Writer combinations, despite substantial performance differences among Writers.

These results suggest that the learned writing practices are not specific to a particular agent or Writer, but transfer across different configurations. The consistent gains further indicate that EvoWriter improves the reusable writing harness itself rather than relying on the capability of a specific underlying model or code harness.

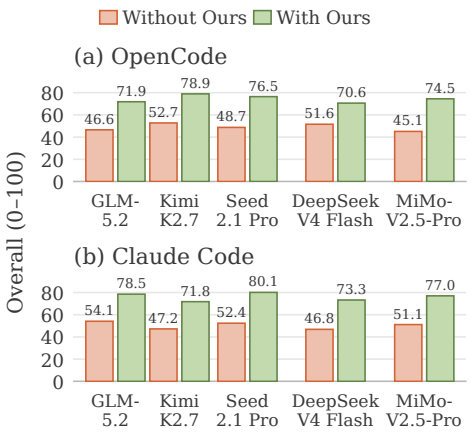



Figure 3: Controlled Writing Evaluation.

## 5.2 Evolution Analysis

We next study what makes writing-harness evolution effective. Specifically, we examine three factors: what guidance should drive evolution, which parts of the harness should be updated, and how evaluator choice affects the assessment of evolved systems. We analyze guidance and update scope on the ten evolution papers in the Evolution Analysis Set.

**Guidance source.** Figure 4A compares three sources of update guidance. *Self* uses the Writer model to evaluate its own writing and direct updates, with the Writer and evaluator kept isolated. *Rubric* uses feedback under predefined writing criteria. Our *Reference* approach compares each draft with the corresponding expert paper and turns the identified gaps into agentic workflow and skill updates. Writing Score averages Quality, Evidence, Retention, and Support.

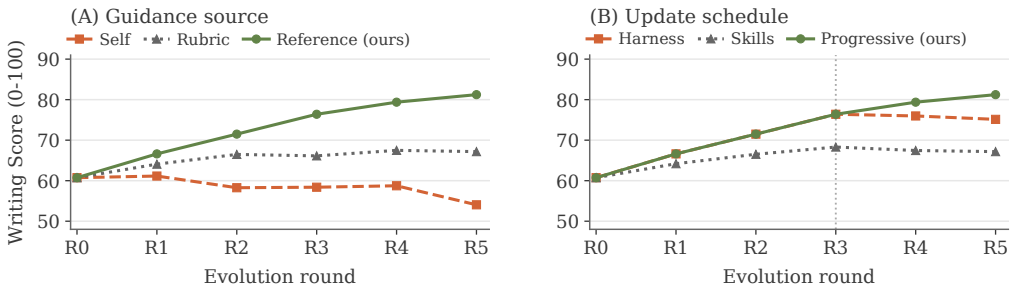



Figure 4: Mean Reading on the Evolution Analysis Set's training split: (A) guidance source; (B) update schedule. Progressive switches to skill-only updates after R3.

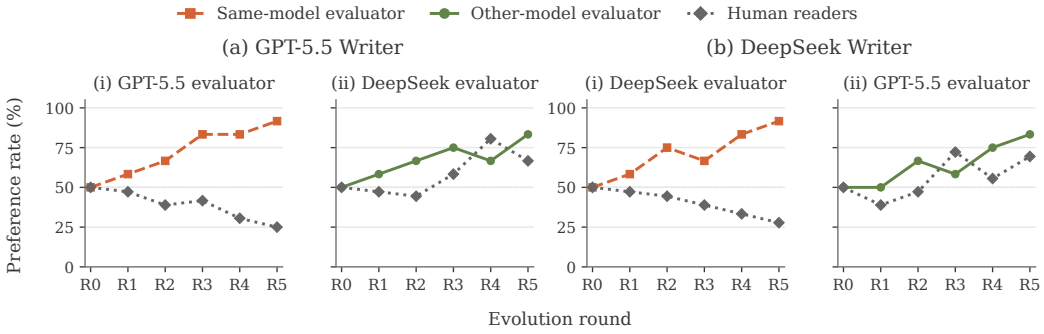



Figure 5: Pairwise preference across evolution rounds for (a) GPT-5.5 and (b) DeepSeek Writers. For each Writer, (i) uses same-model feedback and (ii) uses different-model feedback. Preference is the share favoring $R_k$ over R0, with ties counted as half; R0 is fixed at 50%.

Self yields no net gain over five rounds and ends slightly below initialization. This suggests that self-evaluation alone may reinforce the model's own writing preferences rather than identify changes preferred by human readers. Rubric guidance brings modest improvements by introducing human-defined criteria, but these fixed criteria provide limited information about how good writing is realized in context. Reference guidance achieves the strongest improvement, including before the R3 switch. Expert papers provide concrete examples of how arguments are organized and evidence is presented, allowing the system to learn writing practices beyond those encoded in the initial human-designed harness.

**Evolution schedule.** Figure 4B compares three update schedules under reference guidance. *Harness* updates both workflows and skills throughout evolution, while *Skills* freezes the workflow and updates only writing instructions. *Progressive* begins with full-harness evolution and switches to skill-only updates when the selection score decreases for two consecutive rounds. At the switch, it rolls back to the best-performing harness observed so far and freezes its workflow.

Harness improves faster than Skills in the early rounds, but both peak at R3 and decline afterward. This decline triggers Progressive to roll back to the R3 harness and continue with skill-only updates, leading to further gains and the strongest final result. These results suggest that workflow evolution is useful for establishing a strong writing process early on, while continued workflow changes can become unstable. Switching to skill refinement preserves the learned process while allowing its writing practices to improve further.

**Evaluator identity and pairwise preference.** We examine whether evaluator identity affects the direction of evolution by comparing same-model and different-model feedback. Starting from the same initialization (R0), we track pairwise preference for each evolved version (R1–R5) against R0 using both the model evaluator and independent human readers on the same held-out papers.

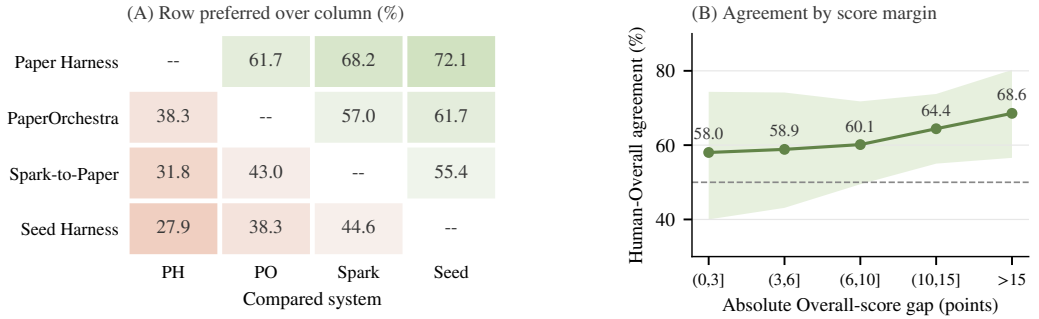



Figure 6: Reader preference and Writing Overall. (A) Each cell gives the row system's preference over the column system, ordered by mean Overall. (B) The share of choices favoring the higher-Overall manuscript, with ties counted half in both panels and 95% resampling intervals in B.

Figure 5 shows a clear divergence under same-model feedback: as evolution proceeds, the model increasingly prefers the evolved writing, while human preference decreases. This suggests that using the Writer itself as the evaluator can gradually steer evolution toward its own writing preferences. The divergence is much smaller when Writer and evaluator use different models, showing that evaluator separation provides a more reliable signal for evolution. Because expert papers provide a reference for preferred scientific writing, we can further validate evaluator quality on held-out papers before using it to guide evolution.

### 5.3 Human Evaluation

**Independent reader preference.** On all 45 held-out test papers, readers compare anonymous complete manuscripts with system identities and automatic scores hidden. As shown in Figure 6A, readers consistently prefer WritingHarness over Seed Harness, PaperOrchestra, and Spark-to-Paper, consistent with the Overall results in the main evaluation. WritingHarness also receives higher ratings for perceived understanding, argument progression, and evidence organization, suggesting that its automatic gains translate into improvements that readers can perceive.

**Relation to Writing Overall.** Figure 6B examines whether differences in Overall scores reflect reader preference. Human–Overall agreement remains above chance across all score margins and generally increases as the score gap widens, from 58.0% for the closest comparisons to 68.6% for gaps above 15 points. This trend suggests that larger Overall differences tend to correspond to quality differences that are more consistently perceived by readers. When two manuscripts receive similar Overall scores, their relative quality may be less clear and reader preference can depend more on finer-grained aspects of the writing. Overall is therefore more informative for distinguishing clear differences in writing quality than for resolving close comparisons.

## 6 Conclusion

We presented Evo to investigate whether research writing should rely on fixed, human-designed workflows or whether better writing harnesses can be learned from expert writing experience. EvoWriter provides an effective way to automatically build and improve a WritingHarness from expert papers, discovering reusable workflows and skills beyond the initial human design. The resulting harness consistently improves research writing across different Writer models, while our analyses show the benefits of expert-paper guidance, evaluator separation, and progressive workflow-to-skill evolution. More broadly, our results suggest that expert examples may provide a general source of experience for automatically constructing stronger task harnesses beyond research writing.

## AI USE STATEMENT

In our experiments, we used generative AI models to produce manuscripts, compare drafts with expert papers, propose workflow and skill updates, and assess manuscript quality (Sections 4 and 5). We also used generative AI to assist with coding and with drafting and revising text in this paper. The authors take responsibility for the final content of this work, including its text, experimental results, and scientific claims.

## ETHICS STATEMENT

EvoWriter and WritingHarness are intended as a writing assistant for human researchers. They generate manuscript drafts for authors to review and revise, with human researchers retaining authorship and responsibility for the final paper. Inaccurate claims can mislead readers, and unattributed reuse of source material can deprive original authors of appropriate credit. Authors should check the manuscript's factual accuracy, originality, scientific claims, and citations before submission. The writing harness checks citations and reviews drafts against the supplied research materials (Section 4.4). Our human evaluation asked participants to compare research manuscripts (Section 5.3). All participants consented to take part, and their personal information was de-identified.

## REPRODUCIBILITY STATEMENT

Section 3 describes the task inputs and dataset splits; Appendix A details dataset composition, material construction, and an example input package. Section 4 presents the evolution procedure, update selection, information-access boundaries, and writing harness. Appendices E and F provide writing-prompt excerpts and document core skill changes. Section 5 describes the compared systems, Writer models, and evaluation measures, with scoring and aggregation rules in Appendix B. Complete evaluation prompts for all six metrics are provided in the accompanying supplementary files. Section 5.2 reports controlled evolution comparisons, Section 5.3 describes the blinded reader-preference evaluation, and Appendix D shows representative generated manuscripts.

## REFERENCES


Anthropic. Claude Code. Official documentation, 2026. Coding agent.

Eric Chamoun, Michael Schlichtkrull, and Andreas Vlachos. Automated Focused Feedback Generation for Scientific Writing Assistance. In *Findings of ACL*, 2024. URL https://aclanthology.org/2024.findings-acl.580/.

Jing Chen, Zhiheng Yang, Yixian Shen, Jie Liu, Adam Belloum, Paola Grosso, and Chrysa Papagianni. SurveyGen-I: Consistent Scientific Survey Generation with Evolving Plans and Memory-Guided Writing. In *Proceedings of IJCNLP-AACL*, 2025. URL https://aclanthology.org/2025.ijcnlp-long.193/.

Ali E. Ghareeb, Benjamin Chang, Ludovico Mitchener, Angela Yiu, Caralyn J. Szostkiewicz, Dmytro Shved, Gavin J. Gyimesi, Jon M. Laurent, Samantha M. Wright, Muhammed T. Razzak, Andrew D. White, Silvia C. Finnemann, Michaela M. Hinks, and Samuel G. Rodriques. A multi-agent system for automating scientific discovery. *Nature*, 2026. URL https://doi.org/10.1038/s41586-026-10652-y.

Gregory Hok Tjoan Go, Khang Ly, Anders Søgaard, Seyed Amin Tabatabaei, Maarten de Rijke, and Xinyi Chen. LiRA: A Multi-Agent Framework for Reliable and Readable Literature Review Generation. In *Proceedings of AAAI*, 2026. URL https://ojs.aaai.org/index.php/AAAI/article/view/41489.

Hengzhi Hu, Qing Zhou, and Harwati Hashim. Negotiating identity in the age of ChatGPT: non-native English researchers' experiences with AI-assisted academic writing. *Humanities and Social Sciences Communications*, 2025. URL https://doi.org/10.1057/s41599-025-05351-4.

Imbad0202. Academic Research Skills for Codex. https://github.com/Imbad0202/academic-research-skills-codex, 2026. Software repository.

Léane Jourdan, Florian Boudin, Richard Dufour, and Nicolas Hernandez. Text revision in Scientific Writing Assistance: A Review. In *Proceedings of BIR*, 2023. URL https://ceur-ws.org/Vol-3617/paper-04.pdf.

Andrej Karpathy. autoresearch. https://github.com/karpathy/autoresearch, 2026. Software repository.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-End Optimization of Model Harnesses. arXiv preprint arXiv:2603.28052, 2026. URL https://arxiv.org/abs/2603.28052.

Jiakun Li, Hui Zong, Erman Wu, Rongrong Wu, Zhufeng Peng, Jing Zhao, Lu Yang, Hong Xie, and Bairong Shen. Exploring the potential of artificial intelligence to enhance the writing of English academic papers by non-native English-speaking medical students - the educational application of ChatGPT. *BMC Medical Education*, 2024. URL https://doi.org/10.1186/s12909-024-05738-y.

Weixin Liang, Yuhui Zhang, Hancheng Cao, Binglu Wang, Daisy Yi Ding, Xinyu Yang, Kailas Vodrahalli, Siyu He, Daniel Scott Smith, Yian Yin, Daniel A. McFarland, and James Zou. Can Large Language Models Provide Useful Feedback on Research Papers? A Large-Scale Empirical Analysis. *NEJM AI*, 2024. URL https://doi.org/10.1056/AIoa2400196.

Weixin Liang, Yaohui Zhang, Zhengxuan Wu, Haley Lepp, Wenlong Ji, Xuandong Zhao, Hancheng Cao, Sheng Liu, Siyu He, Zhi Huang, Diyi Yang, Christopher Potts, Christopher D. Manning, and James Zou. Quantifying large language model usage in scientific papers. *Nature Human Behaviour*, 2025. URL https://doi.org/10.1038/s41562-025-02273-8.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, and Yu-Gang Jiang. Agentic Harness Engineering: Observability-Driven Automatic Evolution of Coding-Agent Harnesses. arXiv preprint arXiv:2604.25850, 2026. URL https://arxiv.org/abs/2604.25850.

Jiaqi Liu, Shi Qiu, Mairui Li, et al. AutoResearchClaw: Self-Reinforcing Autonomous Research with Human–AI Collaboration. arXiv preprint arXiv:2605.20025, 2026. URL https://arxiv.org/abs/2605.20025.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of AI research. *Nature*, 2026. URL https://doi.org/10.1038/s41586-026-10265-5.

Yaxin Luo et al. AutoDesign: Meta-Harness Optimization for Long-Horizon Agentic Design. arXiv preprint arXiv:2608.13560, 2026. URL https://arxiv.org/abs/2608.13560.

OpenAI. Codex. Official software repository, 2026. Coding agent.

Orchestra Research. AI Research SKILLs. https://github.com/Orchestra-Research/AI-Research-SKILLs, 2026a. Software repository.

Orchestra Research. ml-paper-writing. Official SKILL.md, 2026b. Standalone skill, version 1.2.0.

Zhuoyang Qian, Wei Shi, Xu Lin, Li Ling, Meng Luo, Ziming Wang, Zhiwei Zhang, Tengyue Xu, Gaoge Liu, Zhentao Zhang, Shuo Zhang, Ziqi Wang, Zheng Feng, Yan Luo, Shu Xu, Yongjin Chen, Zhibo Feng, Zhuo Chen, Bruce Yuan, Biao Wu, Harry Wang, and Kris Chen. Story2Proposal: A Scaffold for Structured Scientific Paper Writing. arXiv preprint arXiv:2603.27065, 2026a. URL https://arxiv.org/abs/2603.27065.

Zhuoyang Qian, Biao Wu, Yiran Wang, Chris D. Yan, Desan Dai, Liangwei Zheng, Jin Jiang, Junsheng Zhang, and Wenhao Wang. Spark-to-Paper: End-to-End Research Paper Generation as a Composable Skill. arXiv preprint arXiv:2608.11924, 2026b. URL https://arxiv.org/abs/2608.11924.

Yiwen Song, Yale Song, Tomas Pfister, and Jinsung Yoon. Paperorchestra: A multi-agent framework for automated AI research paper writing. In *Proceedings of COLM*, 2026. URL https://openreview.net/forum?id=3PSApLjKex.

Liyan Tang, Cyrus Rashtchian, Chun-Sung Ferng, Andrew Tomkins, Da-Cheng Juan, and Tu Vu. WikiSkill: Compiling Agent Experience into Persistent Knowledge for Skill Evolution. arXiv preprint arXiv:2608.27454, 2026. URL https://arxiv.org/abs/2608.27454.

Yixuan Weng, Minjun Zhu, Guangsheng Bao, Hongbo Zhang, Jindong Wang, Yue Zhang, and Linyi Yang. CycleResearcher: Improving Automated Research via Automated Review. In *Proceedings of ICLR*, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/0a48036026dc7946ef6033ae14719cc5-Abstract-Conference.html.

Yixuan Weng, Minjun Zhu, Qiujie Xie, Qiyao Sun, Zhen Lin, Sifan Liu, and Yue Zhang. DeepScientist: Advancing Frontier-Pushing Scientific Findings Progressively. arXiv preprint arXiv:2509.26603, 2025b. URL https://arxiv.org/abs/2509.26603.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist-v2: Workshop-Level Automated Scientific Discovery via Agentic Tree Search. arXiv preprint arXiv:2504.08066, 2025. URL https://arxiv.org/abs/2504.08066.

Yifan Yang et al. SkillOpt: Executive Strategy for Self-Evolving Agent Skills. arXiv preprint arXiv:2605.23904, 2026. URL https://arxiv.org/abs/2605.23904.

Yuan1z0825. Nature Skills. https://github.com/Yuan1z0825/nature-skills, 2026. Software repository.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-Harness: Harnesses That Improve Themselves. arXiv preprint arXiv:2606.09498, 2026. URL https://arxiv.org/abs/2606.09498.

# A Dataset Details

## A.1 Dataset Composition

Our benchmark contains 100 paper-specific material packages from five research venues: ACL (14), CVPR (11), ICLR (17), ICML (29), and NeurIPS (29). Each package contains a factual research dossier and a bibliography, together with separate evidence files where available. The writing task is to organize these materials into a coherent manuscript in the target conference format.

Figure 7 shows the distribution by venue and primary contribution type. Each paper has one primary type: method (23), analysis (17), theory (12), benchmark (10), dataset (10), survey (10), application (9), or system (9). These categories distinguish different writing demands, from explaining methods and theoretical results to presenting resources, empirical analyses, and literature syntheses.

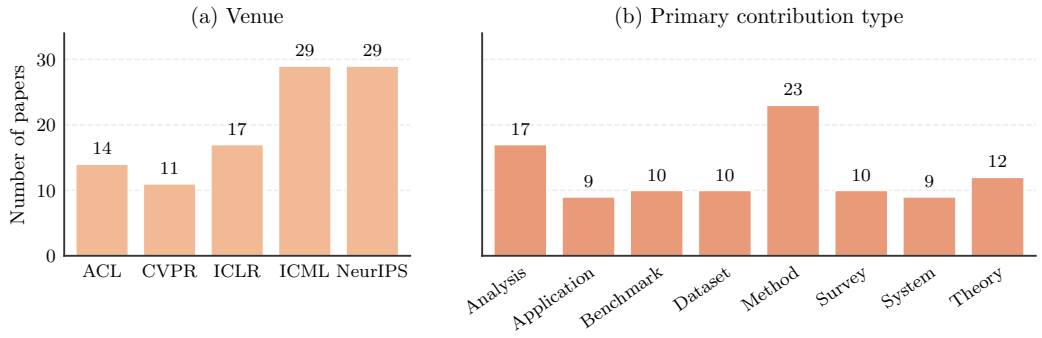



Figure 7: Composition of the 100-paper benchmark by venue and primary contribution type. Each paper is counted once in each panel; labels show absolute counts, which also equal percentages for this corpus.

Table 2 summarizes the material packages. All 100 packages contain a dossier and at least one BibTeX file. Across the corpus, the dossiers average 2,623 whitespace-delimited tokens, with a sample standard deviation of 527. The corpus contains 1,211 evidence files distributed across 99 packages; the median is 11 files per package, with a range of 0–46. The zero corresponds to *High-Dimensional Calibration from Swap Regret*, a purely theoretical paper whose definitions, algorithms, and theoretical results are contained in its dossier; it has no separate empirical data files. These files contain numerical results, experimental settings, theoretical information, and other supporting material, so their count measures the size of the evidence collection rather than the number of experiments.

Table 2: Material-package statistics by venue. Dossier length counts whitespace-delimited tokens, including Markdown and equation markup; SD is the sample standard deviation. Evidence files are separate Markdown data files, not independent experiments.

| Venue | $N$ | Dossier length mean $\pm$ SD | Evidence files median [min, max] |
|---|---|---|---|
| ACL | 14 | 2,955 $\pm$ 81 | 10 [2, 19] |
| CVPR | 11 | 2,978 $\pm$ 73 | 8 [5, 21] |
| ICLR | 17 | 2,596 $\pm$ 546 | 13 [4, 22] |
| ICML | 29 | 2,607 $\pm$ 565 | 14 [3, 46] |
| NeurIPS | 29 | 2,359 $\pm$ 557 | 9 [0$^\dagger$, 28] |
| All | 100 | 2,623 $\pm$ 527 | 11 [0$^\dagger$, 46] |

$^\dagger$The zero corresponds to the purely theoretical paper *High-Dimensional Calibration from Swap Regret* (NeurIPS), whose material package contains no separate evidence files.

### A.2 MATERIAL CONSTRUCTION

We organize source-paper materials using the material-curation procedure in our writing framework. The procedure separates recovering the research record from designing the manuscript's argument: material preparation produces a factual dossier, while the subsequent planning stage determines the story, section structure, and allocation of evidence.

**Recovering the research record.** The curator reads the complete supplied material set and distinguishes source facts, measured results, author interpretations, hypotheses, planned work, negative findings, limitations, and unresolved conflicts. It then organizes the applicable content into a self-contained dossier covering the research problem, method or theoretical object, definitions, experimental conditions, findings, and scope. Equations and technical conditions are retained when needed to understand the evidence. Interpretations and planned experiments remain distinct from established results.

**Separating evidence from prose.** Tables, logged measurements, and other useful evidence are extracted into semantically named Markdown files. An evidence file preferably contains one coherent body of information; material with different datasets, splits, protocols, metric families, or model configurations is separated when those distinctions affect interpretation. The extraction preserves units, metric directions, uncertainty, seeds, and distinctions between exact and approximate values. Bibliographic material is retained as a separate input component.

**Preserving evidence boundaries.** Permitted transformations include format normalization, joining explicit fragments of the same table, separating mixed records, and removing exact duplicates. Missing values, baselines, and unexecuted experiments are not completed from inference. When apparently conflicting observations refer to explicitly different conditions, both are retained with those conditions; otherwise, the ambiguity remains unresolved. Negative results and limitations are preserved rather than removed during condensation. A final pass checks that the dossier and evidence files remain grounded in the source materials and have not become a persuasive rewrite of the paper.

The released material packages use `draft.md` for the dossier and `datas/*.md` for evidence files. In a writing run, these components are placed under `inputs/draft/` and `inputs/data/`, respectively. Appendix E.1 provides the main material-preparation instructions.

### A.2.1 EXAMPLE MATERIAL PACKAGE

**Research dossier: draft.md**

```
# LLMs Get Lost In Multi-Turn Conversation

## Motivation and research scope

The study examines whether conversational LLMs can solve a task when
↪  users reveal its requirements gradually. Real users often begin
↪  with incomplete requests, whereas most benchmarks supply complete
↪  instructions in a single turn. Existing multi-turn evaluations
↪  may be episodic, with each turn treated as an independent
↪  subtask.

The target setting instead requires integrating constraints across
↪  turns: a later clarification can affect the entire solution. The
↪  contribution is an empirical evaluation framework for this
↪  setting. It distinguishes task-solving ability from consistency
↪  across conversations with the same underlying requirements.

## Method and comparison conditions

A fully specified instruction is split into shards: an initial
↪  high-level goal and later, decontextualized clarifications.
↪  Preparation combines automatic segmentation and rephrasing,
↪  verification using concatenated and reordered shards, and manual
↪  inspection and editing. The shards must preserve the information
↪  needed to solve the original task.

- Full: give the original complete instruction in a single turn.
- Concat: combine all shards into one prompt, retaining the rewritten
↪  wording while making every requirement available at once.
- Sharded: reveal the shards gradually over a conversation, with at
↪  most one new piece of information per user turn.

Full versus Concat checks the effects of rewriting; Concat versus
↪  Sharded compares simultaneous and gradual delivery of the
↪  rewritten information.

A simulated user selects clarifications using the remaining shards
↪  and conversation history. The assistant responds freely, without
↪  a prescribed clarification strategy. Extracted answers receive
↪  task-specific scores. The conversation stops after a correct
↪  answer or when no shards remain at a new turn's start.

## Evaluation setup

The main tasks are code generation, database querying, API calling,
↪  mathematics, data-to-text generation, and summarization. Math
↪  uses GSM8K with exact match of standardized numerical answers.
↪  For each model, the same instruction is evaluated repeatedly
↪  under each delivery condition. Mean performance summarizes the
↪  average outcome. Aptitude is the 90th-percentile score;
↪  unreliability is the gap between the 90th- and 10th-percentile
↪  scores. These measures distinguish occasional success from
↪  consistent performance across conversations for the same task.

## Main findings

Across the tested models and main tasks, Sharded performance is lower
↪  than Full, while Concat generally remains close to Full.
Strong single-turn models and reasoning models also exhibit the
↪  decline.
```

**Research dossier: draft.md (continued)**

```
Best-case aptitude decreases, but the larger deterioration is in
↪  reliability: outcomes vary substantially across conversations for
↪  the same instruction.

Qualitative analyses identify premature answer attempts, continued
↪  revision of earlier wrong answers, uneven use of information
↪  across turns, and increasingly verbose responses.

## Author interpretation

The authors interpret the Concat comparison as evidence that
↪  rewriting alone does not explain the performance loss. They
↪  propose that early assumptions persist in later answers and
↪  hinder recovery. This is a proposed explanation of the observed
↪  behaviors, not an established sole cause. Their practical
↪  implication is that multi-turn reliability requires evaluation
↪  and improvement beyond single-turn task competence.

## Negative results and limitations

Consolidating requirements in a final recap or repeating them across
↪  turns partially improves performance but does not restore the
↪  single-turn baseline. Lowering temperature does not eliminate
↪  multi-turn unreliability. An incremental translation control
↪  shows no degradation; the authors attribute this exception to its
↪  decomposable, sentence-level structure.

The evaluation uses English, text-based analytical tasks and
↪  automated users. Its structured delivery eventually supplies
↪  sufficient information and does not capture all human
↪  misunderstandings, frustration, or abandonment. The findings
↪  therefore concern underspecified tasks with interacting
↪  requirements, rather than all multi-turn interaction.
```

The corresponding evidence file, `datas/six-task-conversation-performance.md`, stores the measured performance separately. Table 3 reproduces its Math columns for two models, with the original model labels and numerical values. The dossier identifies Math as GSM8K, evaluated by exact match of standardized numerical answers on a 0–100 scale. Keeping these values separate from the experimental definitions lets a writer recover both the result and the conditions needed to interpret it.

**Evidence file: six-task-conversation-performance.md (excerpt)**

Table 3: An actual evidence-file excerpt from the ICLR material package: Math performance under three instruction-delivery settings. Only two model rows and the three Math columns are shown; values are copied without recalculation.

| Model | Full | Concat | Sharded |
|---|---|---|---|
| OpenAI 4o-mini | 88.1 | 88.0 | 58.7 |
| OpenAI 4o | 93.8 | 91.9 | 67.9 |

# B Evaluation Metrics and Protocols

This appendix specifies the six component metrics in Table 1, their aggregation into Writing Overall, and the separate acceptance statistic. The complete LLM-as-a-judge prompts for all six metrics are provided in the accompanying supplementary files. They assess completed manuscripts rather than guide writing. All judgments below are produced by LLMs; Section 5.3 reports the separate human study.

Clarity and Presentation use the AI Scientist-v2 reviewer (Yamada et al., 2025). Quality assesses how readily readers understand the paper, recognize its motivation, follow its narrative, remain engaged, and retell its central idea on a first reading.

## B.1 Metric Calculations and Aggregation

Scores are first calculated for each manuscript; the reported system scores are paper-level macro-averages on the fixed 45-paper test cohort.

**Clarity and Presentation.** For manuscript $i$, let $C_i$ and $P_i$ be the validated `Clarity` and `Presentation` fields from the AI Scientist-v2 review (Yamada et al., 2025). Each is an integer from 1 to 4 for a single review. Clarity covers explanation, organization, and information available to the reader. Presentation covers writing style, clarity of presentation, and context relative to prior work. The reviewer receives extracted manuscript text, so this Presentation score is not a visual-layout assessment.

**Quality.** Quality is the weighted score from the first-reading rubric in Appendix B.3. For AI judgment $j$ of manuscript $i$, let $u_{ij}, m_{ij}, n_{ij}, h_{ij}$, and $t_{ij}$ denote ease of understanding, perceptibility of motivation, narrative coherence and progression, reading interest, and central impression and retellability. Each dimension uses a 1–10 scale. The weighted absolute score and its within-paper aggregate are

$$\begin{aligned} q_{ij} &= 0.30u_{ij} + 0.20m_{ij} + 0.20n_{ij} + 0.15h_{ij} + 0.15t_{ij}, \\ Q_i &= \frac{1}{|\mathcal{J}_i|} \sum_{j \in \mathcal{J}_i} q_{ij}. \end{aligned} \tag{6}$$

Here $\mathcal{J}_i$ is the prescribed set of model/order judgments, not a set of independent papers. The Quality comparison protocol pairs a generated manuscript with its corresponding expert original and uses DeepSeek-V4-Flash and GPT-5 in both presentation orders. Anonymous labels are mapped back to manuscripts before averaging the four absolute scores. A pairwise win is not an absolute Quality score. For an expert original appearing in several comparisons, its repeated scores are first aggregated within that paper before entering the cohort average.

**Evidence.** Evidence measures argument organization using the rubric in Appendix B.4. Each isolated evaluation $j$ assigns four integer scores from 1 to 5: claim progression $a_{ij}$, evidence timing $b_{ij}$, transitions $c_{ij}$, and closure $d_{ij}$. With the number $K_i$ of evaluations fixed before scoring,

$$E_i = \frac{1}{K_i} \sum_{j=1}^{K_i} \frac{a_{ij} + b_{ij} + c_{ij} + d_{ij}}{4}. \tag{7}$$

Scores and cited evidence are validated before averaging. The repeat count $K_i$ is a run setting not specified by the prompt. Unlike Evidence, Support assesses factual grounding rather than argument organization.

**Retention.** Retention is main-text recall derived from the observed evidence locations in Appendix B.5. Let $U_i$ be the frozen universe of evidence units actually supplied to the writer, $G_i^{\text{main}} \subseteq U_i$ the independently annotated units substantively presented in the source main text, and $S_i^{\text{main}} \subseteq U_i$ the units substantively presented in the generated main text. Then

$$R_i = 100 \, \frac{|S_i^{\text{main}} \cap G_i^{\text{main}}|}{|G_i^{\text{main}}|}. \tag{8}$$

The denominator counts supplied units from the source main text. Recovering six of eight such units gives $R_i = 75\%$. Appendix-only presentation receives no main-text recall credit. Original (GT) has 100% by construction because its selected set is $G_i^{\text{main}}$ itself, whenever that set is nonempty.

**Support.** Support is derived from the claim-level verification procedure in Appendix B.6. Let $\mathcal{C}_i$ be the frozen, audited, deduplicated inventory of substantive atomic claims, and let $\ell_i(c)$ be the final evidence-grounded label for claim $c$. For a completed assessment,

$$F_i = \begin{cases} 100\,\dfrac{\sum_{c\in\mathcal{C}_i} \mathbf{1}\{\ell_i(c) = \texttt{supported}\}}{|\mathcal{C}_i|}, & |\mathcal{C}_i| > 0, \\ 0, & |\mathcal{C}_i| = 0 \text{ after complete inspection.} \end{cases} \tag{9}$$

Unsupported and contradicted claims remain in the denominator. Partial support does not count as full support. Missing or unreadable inputs do not qualify for the zero-claim convention and instead leave the score incomplete.

**Writing Overall and Acceptance.** Using the native component scales above, the equal-weight aggregate is

$$O_i = \frac{100}{6}\left[\frac{C_i - 1}{3} + \frac{P_i - 1}{3} + \frac{Q_i - 1}{9} + \frac{E_i - 1}{4} + \frac{R_i}{100} + \frac{F_i}{100}\right]. \tag{10}$$

For the fixed cohort $\mathcal{I}$, $N = |\mathcal{I}| = 45$, each reported component and Overall is a macro-average,

$$\overline{X} = \frac{1}{N}\sum_{i\in\mathcal{I}} X_i, \qquad X \in \{C, P, Q, E, R, F, O\}. \tag{11}$$

Thus Support and Retention are not pooled over claims or evidence units across papers. If $D_i$ is the manuscript's validated reviewer decision,

$$\text{Acceptance} = \frac{100}{N}\sum_{i\in\mathcal{I}} \mathbf{1}\{D_i = \texttt{Accept}\}. \tag{12}$$

Acceptance is an AI recommendation rate, not an observed venue decision, and is excluded from Overall. Overall and Acceptance require no additional scoring prompt.

**Validity and missing evaluations.** Calculations retain full precision until display rounding. Missing components are not assigned zero, Overall weights are not redistributed, and incomplete papers are not silently excluded from the fixed cohort. For repeated reviews, score aggregation and the rule producing one decision per paper must be fixed before evaluation.

### B.2 Shared Reviewer: Clarity, Presentation, and Acceptance

Three message components form one AI Scientist-v2 review call (Yamada et al., 2025). The system message sets a strict reviewing stance. The user message combines the complete review form, any configured original few-shot examples, and the manuscript-input wrapper, replacing `{text}` with the full extracted manuscript. The review is parsed and validated before selecting `Clarity`, `Presentation`, and `Decision`.

The native `Quality` (1–4) and `Overall` (1–10) fields remain in the original form but do not supply our Quality or Writing Overall scores. Exact few-shot examples and execution settings are not specified by these templates and are required to reproduce individual review calls.

### B.3 Quality: First-Reading Evaluation

The supplementary prompt evaluates how readily a scientifically literate reader can follow and understand each manuscript in its original order. It takes two complete, anonymously labeled papers, with inspectable pages for visual judgments, and returns five dimension scores, a weighted absolute score for each paper, and a separate preference. Equation 6 gives the aggregation. Scientific novelty, correctness, and factual support are outside its scope.

### B.4 EVIDENCE: ORGANIZATION OF CLAIMS AND EVIDENCE

The supplementary prompt specifies the four-dimension Evidence rubric. It accepts the complete manuscript, including displays, captions, and appendices, and returns four ratings with locators, excerpts, and explanations. Its scope includes empirical experiments, theoretical derivations, and synthesis in surveys. Equation 7 gives the equal-weight calculation.

### B.5 RETENTION: MAIN-TEXT EVIDENCE RECALL

The experiment-placement prompt extracts observed locations; software then computes Equation 8. Both manuscripts use the same complete Markdown extraction format and include appendices. The third input is the fixed candidate list with its original atomic IDs. Gold location labels and review records are not supplied to this evaluation call.

**Candidate universe and frozen Gold.** The candidate universe $U_i$ includes only evidence units actually provided to the writing system from both source main text and appendix. Unit boundaries and IDs are fixed before examining generated manuscripts. The normal writing task and main-text budget determine selection; the writer receives neither source-location labels nor evaluation Gold.

An AI annotator receives only the original paper and $U_i$, records locations, dual-location presentations, and supporting source evidence, and constructs disjoint sets $G_i^{\text{main}}$ and $G_i^{\text{app}}$ covering $U_i$. An isolated AI reviewer checks a saved random sample containing at least one item from each nonempty location class, together with every ambiguous, dual-location, or partial-coverage case. It receives sampled IDs and source material, but not initial labels, generated papers, or evaluator predictions. A separate AI adjudication step resolves disagreements against the source evidence. Unresolved cases remain incomplete. The reviewed Gold is fixed before scoring. The original-paper sets extracted by the placement prompt are checked against this independent Gold.

**Content matching and location rules.** Substantive presentation can be a table, figure, or textual result report. Matching uses content, conditions, comparisons, and results, not display numbers alone. When a unit is substantively presented in both places, its scored label is `main_text`, with both locations recorded separately. A pointer such as "see Appendix Table A.1" is not a main-text presentation. Confirmed absence is not appendix placement. All scored IDs must belong to $U_i$; after resolving ambiguities, the generated main-text, appendix-only, and confirmed-missing sets must be disjoint and cover $U_i$.

Unreadable inputs, uncertain identity, partial matching, and ambiguous main/appendix boundaries remain unresolved rather than being scored as omissions. A pair with unresolved issues receives no complete score. When $G_i^{\text{main}}$ is empty, recall is undefined; a predetermined eligibility rule and explicit reporting are required instead of silently changing the cohort. Precision and placement F1 may be retained as diagnostics, since recall alone does not penalize over-selection, but neither replaces the main-table Retention value.

**Non-empirical papers.** The main-table protocol extends evidence units to results or proof-support units in theory papers and synthesis or comparison units in surveys. The same set formula applies to those frozen units. The supplementary template uses experimental terminology; exact wording adaptations for non-empirical units and the separate Gold-annotation instructions are not included in the prompt files. Those instructions are needed to reproduce the extension; the empirical template alone does not verify it.

### B.6 SUPPORT: ATOMIC-CLAIM VERIFICATION

Support uses three templates: claim inventory, evidence verification, and disagreement adjudication. The inventory template runs in `extract` and `audit` modes; two isolated verification calls assess the frozen claims, and adjudication resolves disputed labels. Software validates the outputs and calculates the final score.

**Inventory and denominator.** The extractor identifies substantive claims about the study's methods, empirical or theoretical results, and conclusions throughout the main text, appendices, figures,

tables, and captions. Compound assertions are split into independently checkable propositions, while datasets, conditions, numerical values, comparison targets, and qualifications are preserved. Semantically equivalent repetitions are deduplicated and all occurrences retained. Different scopes are not duplicates, and one sentence or table cell need not equal one claim. General background, subjective value judgments, standalone novelty judgments, unimplemented future plans, and nonfactual transitions are excluded with recorded examples. Apparent lack of support never justifies excluding a substantive claim.

An isolated inventory audit checks completeness, exclusions, duplicates, atomicity, and scope without seeing support labels or target scores. The controller validates coverage and freezes the corrected inventory before verification.

**Verification and adjudication.** Two AI verifiers independently assess the same frozen claims against the same author materials and evidence index; neither sees the other's output. A claim is `supported` only if all substantive content and qualifications follow from the supplied evidence. Partial support or overgeneralization is `unsupported`; conflict with the materials is `contradicted`; missing or unreadable essential inputs produce `unresolved_input`. Simple deterministic deductions are permitted with explicit evidence, but speculation and mere repetition of a conclusion are not supporting results.

The adjudicator resolves disputed labels by rechecking evidence, not by averaging labels or preferring a model's confidence. The controller combines adjudicated and agreed labels and computes Equation 9. Any unresolved input or blocking inventory issue prevents a complete score, even if both verifiers agree that the input is unresolved. Genuine inventory errors require a recorded correction, a newly frozen version, and renewed verification of affected claims; difficult claims are not silently removed. Isolated contexts do not establish statistically independent model errors.

## C Human Evaluation Details

**Comparison protocol.** We compare four systems: Paper Harness, PaperOrchestra, Spark-to-Paper, and Seed Harness. Each system produces one manuscript for each of 45 papers, yielding 180 manuscripts and $\binom{4}{2} = 6$ within-paper system pairs per paper. Each pair is assigned to three readers for separate judgments, giving $45 \times 6 \times 3 = 810$ planned judgments. After excluding 32 skipped judgments, 778 valid judgments remain. Each judgment corresponds to one reader's preference between two manuscripts for the same paper; thus, 778 denotes the number of valid judgments, rather than the number of unique readers or independent papers.

**Agreement with Overall scores.** For each manuscript pair, we identify the higher-scored manuscript using the two manuscripts' per-paper Overall scores, rather than system-level average scores. We then compare this score ordering with the reader's preference. A judgment receives an agreement value of 1 if the reader prefers the higher-scored manuscript, 0.5 for a tie, and 0 if the reader prefers the lower-scored manuscript. Comparisons with equal Overall scores are excluded from this analysis; no such comparisons occur in the evaluated data. Let $N_{\text{high}}$, $N_{\text{tie}}$, and $N_{\text{low}}$ denote the respective judgment counts. Agreement is calculated as

$$\text{Agreement} = \frac{N_{\text{high}} + \frac{1}{2}N_{\text{tie}}}{N_{\text{high}} + N_{\text{tie}} + N_{\text{low}}} \times 100\%. \tag{13}$$

This statistic measures directional agreement between reader preferences and Overall scores, rather than inter-reader agreement.

**Pooled and score-gap results.** Across all valid judgments, readers prefer the higher-scored manuscript in 423 cases, report a tie in 136 cases, and prefer the lower-scored manuscript in 219 cases. The pooled agreement is therefore

$$\frac{423 + 136/2}{778} \times 100\% = \frac{491}{778} \times 100\% = 63.1\%. \tag{14}$$

To examine agreement across score differences, we group judgments by the absolute difference between the two manuscripts' Overall scores, $\Delta = |s_a - s_b|$. Table 4 reports the counts and

agreement for each bin. Figure 6B applies Equation 13 separately within these same bins. For example, the first bin has agreement $(40 + 14/2)/81 \times 100\% = 58.0\%$. The five plotted points and the pooled result therefore use the same 778 judgments. The pooled result is equivalent to weighting the unrounded bin-level agreement rates by their judgment counts, rather than averaging the five rates equally.

Table 4: Human preference outcomes by absolute Overall score gap. High and Low denote preferences for the higher- and lower-scored manuscripts, respectively. Ties receive half credit when computing agreement. All percentages are rounded to one decimal place.

| Score gap $\Delta$ | High | Tie | Low | Judgments | Agreement |
|---|---|---|---|---|---|
| $0 < \Delta \leq 3$ | 40 | 14 | 27 | 81 | 58.0% |
| $3 < \Delta \leq 6$ | 37 | 19 | 23 | 79 | 58.9% |
| $6 < \Delta \leq 10$ | 105 | 33 | 64 | 202 | 60.1% |
| $10 < \Delta \leq 15$ | 120 | 46 | 56 | 222 | 64.4% |
| $\Delta > 15$ | 121 | 24 | 49 | 194 | 68.6% |
| **Total** | **423** | **136** | **219** | **778** | **63.1%** |

# D PAPER VISUALIZATION

We present three illustrative manuscripts generated from paper-specific research materials: two produced by our framework and one produced by a single-agent baseline. Each overview presents the main text and bibliography in their original order, excluding appendices and checklists. Enlarged pages show how the generated manuscripts combine technical explanation with figures and tables. Full-resolution copies of all three main-text and bibliography selections accompany this appendix.

## D.1 SINGLE-AGENT EXAMPLE: LLMS GET LOST IN MULTI-TURN CONVERSATION

This example is the output of the single-agent baseline,which uses codex and gpt-6-astra as the runtime config.

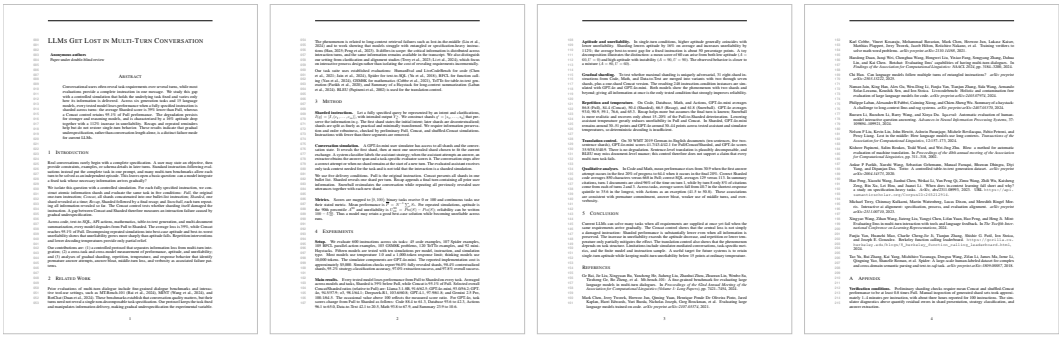



Figure 8: **Single-agent example, complete main text and bibliography.** Pages 1–4 of the compiled single-agent output

### D.2 NeurIPS Example: Auto-Compressing Networks

This example develops an architectural change through a method description, gradient-based diagnostics, and empirical comparisons. The complete output is shown through its 13 main-text pages and two bibliography pages.

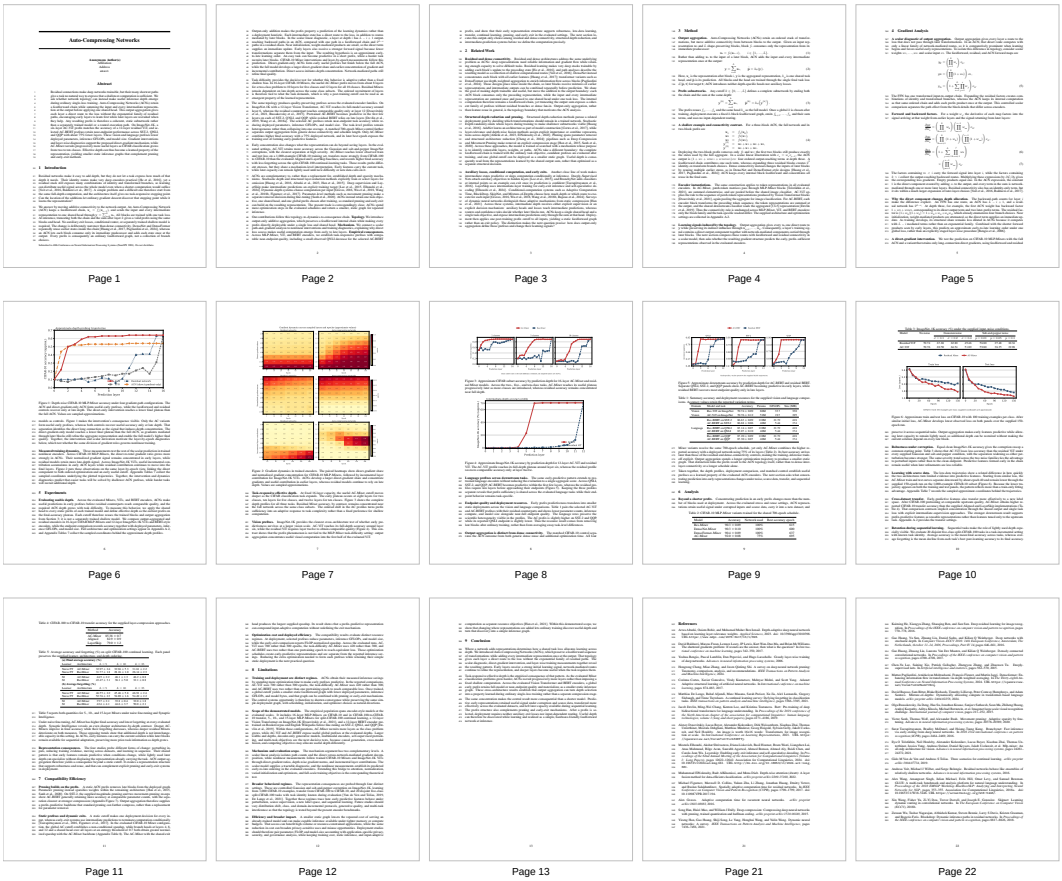



Figure 9: **NeurIPS example, main text and bibliography.** Original pages 1–13 and 21–22 of the generated *Auto-Compressing Networks* manuscript, in reading order from left to right and top to bottom. Page labels refer to the original PDF. Appendix and checklist pages are omitted. The displayed pages preserve their original content and layout.

**NeurIPS example: selected pages.** The larger views below show the manuscript's use of gradient heatmaps and task-specific performance curves. Their surrounding text is retained so that the visual evidence can be inspected together with its explanation.

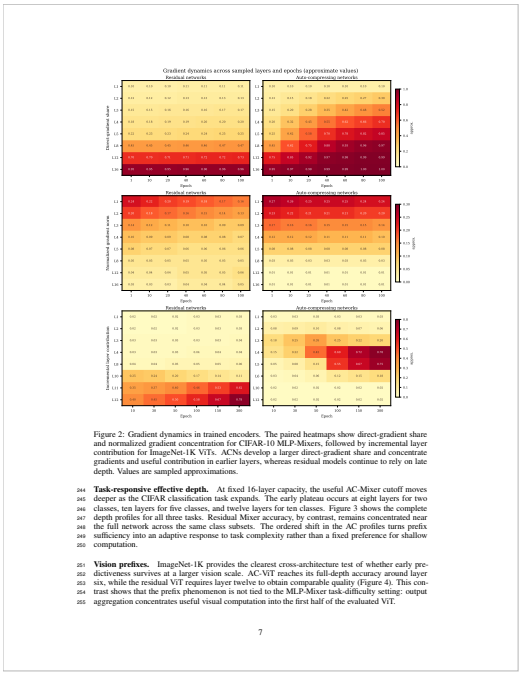


Figure 2: Gradient dynamics in trained encoders. The paired heatmaps show direct-gradient share and normalized gradient concentration for CIFAR-10 MLP-Mixers, followed by incremental layer contribution for ImageNet-1K ViTs. ACNs develop a larger direct-gradient share and concentrate gradients and useful contribution in earlier layers, whereas residual models continue to rely on late depth. Values are sampled approximations.

**Task-responsive effective depth.** At fixed 16-layer capacity, the useful AC-Mixer cutoff moves deeper as the CIFAR classification task expands. The early plateau occurs at eight layers for two classes, ten layers for five classes, and twelve layers for ten classes. Figure 3 shows the complete depth profiles for all three tasks. Residual Mixer accuracy, by contrast, remains concentrated near the full network across the same class subsets. The ordered shift in the AC profiles turns prefix sufficiency into an adaptive response to task complexity rather than a fixed preference for shallow computation.

**Vision prefixes.** ImageNet-1K provides the clearest cross-architecture test of whether early predictiveness survives at a larger vision scale. AC-ViT reaches its full-depth accuracy around layer six, while the residual ViT requires layer twelve to obtain comparable quality (Figure 4). This contrast shows that the prefix phenomenon is not tied to the MLP-Mixer task-difficulty setting: output aggregation concentrates useful visual computation into the first half of the evaluated ViT.

7

Page 7

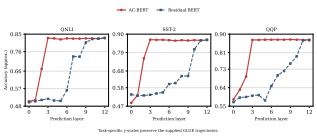


Figure 5: Approximate downstream accuracy by prediction depth for AC-BERT and residual BERT. Separate QNLI, SST-2, and QQP panels show AC-BERT becoming predictive in early layers, while residual BERT recovers most endpoint quality only in late layers.

Table 1: Summary accuracy and deployment resources for the supplied vision and language comparisons. Accuracy values retain the reported variation terms.

| Domain | Model and task | Accuracy | Params | GFLOPs | Size (MB) |
|---|---|---|---|---|---|
| Vision | Res-ViT on ImageNet | 70.74 ± 0.09 | 86M | 33.7 | 330 |
| Vision | AC-ViT on ImageNet | 70.76 ± 0.12 | 51M | 19.7 | 195 |
| Language | Res-BERT on SST-2 | 86.63 ± 0.09 | 110M | 21.72 | 418 |
| Language | AC-BERT on SST-2 | 86.68 ± 0.06 | 46M | 5.44 | 174 |
| Language | Res-BERT on QNLI | 83.14 ± 0.07 | 110M | 21.72 | 418 |
| Language | AC-BERT on QNLI | 83.07 ± 0.10 | 46M | 5.44 | 174 |
| Language | Res-BERT on QQP | 87.20 ± 0.09 | 110M | 21.72 | 418 |
| Language | AC-BERT on QQP | 87.30 ± 0.07 | 46M | 5.44 | 174 |

Mixer variants receive the same 700-epoch schedule, yet only AC-Mixer combines the higher reported accuracy with a deployed network using 75% of its layers (Table 2). Its best accuracy arrives later than those of the residual and dense-connectivity controls, making the training–inference tradeoff explicit. Output aggregation spends a longer optimization trajectory to produce a smaller static graph. That distinction links the prefix result to the ACN topology itself, rather than to dense inter-layer connectivity or a longer schedule alone.

Taken together, the depth profiles, deployment comparison, and matched control establish useful prefixes as a learned property of the evaluated ACN encoders. The next section asks what concentrating prediction into early representations changes under noise, scarce data, transfer, and sequential learning.

## 6 Analysis

**Beyond a shorter prefix.** Concentrating prediction in an early prefix changes more than the number of blocks used at deployment. Across the evaluated stress and reuse settings, ACN representations retain useful signal under corrupted inputs and scarce data, carry it into a new dataset, and

Table 2: CIFAR-10 MLP-Mixer variants trained for the shared 700-epoch schedule.

| Model | Accuracy | Network used | Best accuracy epoch |
|---|---|---|---|
| Res-Mixer | 90.3 ± 0.09 | 100% | 615 |
| DenseNet-Mixer | 90.3 ± 0.10 | 100% | 600 |
| DenseFormer-Mixer | 90.4 ± 0.09 | 100% | 637 |
| AC-Mixer | 92.0 ± 0.08 | 75% | 695 |

9

Page 9

Figure 10: **NeurIPS example, enlarged pages 7 and 9.** The selected pages present mechanism-oriented diagnostics and empirical results. Both pages are reproduced directly from the generated PDF without editing their prose, captions, figures, or numerical values.

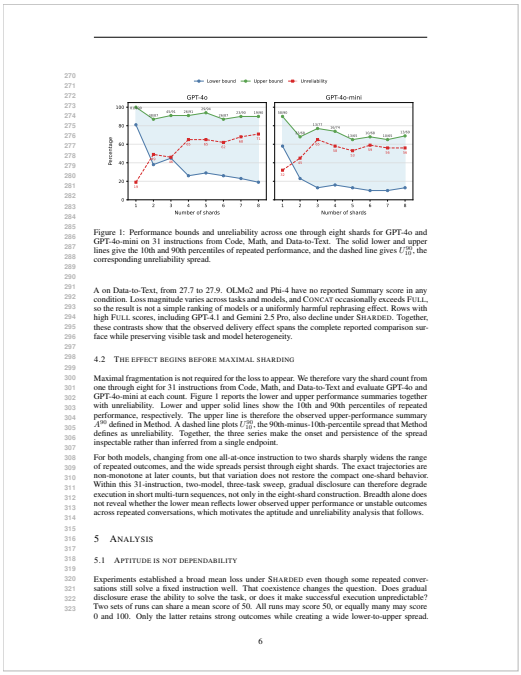
### D.3 ICLR Example: LLMs Get Lost in Multi-Turn Conversation

This example organizes an empirical study of how language-model performance changes when task requirements are supplied across multiple turns. Its main text and bibliography span 15 pages and combine experimental definitions, result tables, plots, and analysis of the observed behavior.

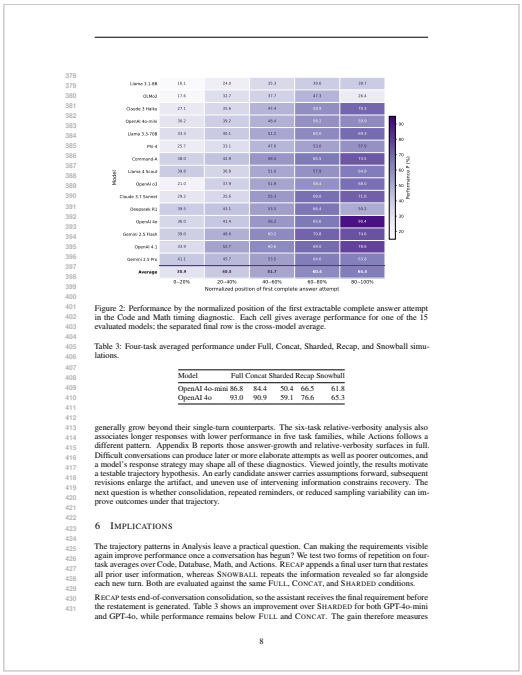



Figure 11: **ICLR example, main text and bibliography.** Pages 1–15 of the generated *LLMs Get Lost in Multi-Turn Conversation* manuscript. Pages retain their original order; page 15 is cropped after the final reference to exclude the appendix that begins on the same page. Labels identify the original PDF pages.

**ICLR example: selected pages.** The enlarged views retain the result displays and the prose that interprets them, illustrating how the framework presents an analysis-oriented paper.



Page 6



Page 8

Figure 12: **ICLR example, enlarged pages 6 and 8.** The generated manuscript combines graphical evidence, a numerical table, and explanatory prose. These are unmodified pages of our output.

# E PROMPTS

We present key excerpts from the principal instruction templates in the `harness` implementation. The excerpts retain the source wording; `[...]` marks an omitted block. Complete copies of these seven templates accompany this appendix. At runtime, the framework combines role instructions with task-specific inputs, output contracts, and project-local writing skills. The listings below expose the central behavioral instructions rather than repeating the path and bookkeeping fields appended to every task.

## E.1 MATERIAL PREPARATION

The material-preparation prompt turns source materials into a factual dossier and separate evidence files before manuscript planning. It preserves the distinction between observations, interpretations, and unresolved questions.

**Source:** `model_material_preparation.md`

```
Create `inputs/draft/draft.md` as a factual research dossier, not
↪  manuscript prose. Include only applicable material supported by
↪  the provided inputs: research problem, intended scope, method or
↪  system, experimental setup, findings, ablations, diagnostics,
↪  negative results, supported claims, limitations, and open
↪  questions. Preserve useful technical detail while keeping source
↪  facts, author interpretations, hypotheses, TODOs, and unresolved
↪  questions distinct.

When useful, convert explicit tables, CSV/TSV data, or logged metrics
↪  into semantically named Markdown files under `inputs/data/`.
↪  Preserve source splits, configurations, units, seeds, uncertainty,
↪  and metric meaning. Do not guess missing values, invent baselines,
↪  derive unsupported metrics, mix incompatible protocols, or select
↪  flattering checkpoints or seeds.

Do not modify `inputs/raw/`, `inputs/template/`, `inputs/skills/`, or
↪  `inputs/manifest.json`. Do not use network search or external
↪  sources. Do not create paper plan artifacts, choose the final
↪  paper story, or write manuscript sections. If supplied materials
↪  conflict and no safe resolution is explicit, omit the disputed
↪  assertion and record the ambiguity in `known_issues`.
```

## E.2 PAPER PLANNING

The planner assigns reader questions and section responsibilities around a central research question. Its outputs guide later writers; it does not draft the manuscript or choose runtime orchestration settings.

**Source:** `model_plan.md`

```
Create the planning artifacts only; do not create tasks, start
↪  subagents, decide retries, or write manuscript prose.
Write all required plan artifacts before writing the SubagentResult.

[...]

Planning requirements:
- Build a reader-facing story spine from the user goal and draft.
- Define section order, reader questions, one-sentence answers,
↪  section jobs, dominant support, and transitions.
- Without changing any artifact schema, give every section enough
↪  positive
```

```
  narrative guidance for a writer to infer: the section dilemma or
  ↪ reader
  tension, the central hinge or discovery, the method/evidence
  ↪ consequence, the
  primary evidence, and the expected change in reader understanding.
  ↪ Put this
  guidance in the existing `Narrative Development` and
  `Section-by-Section Momentum` headings of `story_plan.md`, and
  ↪ project it
  through existing `reader_question`, `one_sentence_answer`,
  ↪ `section_job`,
  transition, and optional `prose_obligations` fields. Do not add new
  ↪ required JSON fields.

[...]

- Before writing `section_plan.json`, identify the central scientific
↪ question
  expressed in `Core Story` and use it as the semantic anchor for the
  ↪ Abstract
  and Introduction. When that question is already clear, the Abstract
  `reader_question` should reuse it or closely paraphrase it. The
  ↪ Introduction
  may motivate or sharpen the same question, but should preserve its
  ↪ scientific
  subject, hypothesized relation, and sought outcome. Treat
  ↪ `reader_question`
  as the unresolved question organizing the section, not as a synopsis
  ↪ of its
  coverage or eventual findings. Put the hinge discovery, empirical
  ↪ answer, and
  payoff in `one_sentence_answer`, `section_job`, and
  ↪ `prose_obligations`.
  Before finishing, compare both front-matter questions with `Core
  ↪ Story` and
  repair any semantic drift.
```

### E.3 Figure and Table Materialization

The display prompt selects a form for each evidence presentation before rendering it. Figures and tables retain the planned question, source evidence, and intended takeaway.

**Source:** `model_display.md`

```
Mandatory skill reading before materialization:
- Before any display-materialization work, read the complete contents
↪ of:
  - `inputs/skills/oral-figure/SKILL.md`
  - `inputs/skills/oral-table/SKILL.md`
  - `inputs/skills/oral-evidence-display/SKILL.md`
- Read `inputs/skills/oral-evidence-display/references/display-form-s⌋
↪ election.md`
  completely before resolving any artifact form.
- For each intent, inspect its real sources and `preserve` list,
↪ compare at least
  two plausible artifact forms, then write
  `paper/displays/display_resolution.json` before materializing files
  ↪ or
  catalogs. One intent produces exactly one formal artifact and one
  ↪ label;
  multi-panel layouts remain one artifact.

[...]
```

```
Display materialization requirements:
- Preserve the reviewer question, source data, supporting claim,
↪  required
  takeaway, `preserve` list, owner, placement, and drop policy from
  `display_plan.json`; do not copy or rewrite these semantic fields in
  ↪  the
  resolution.
- Generate or repair main-text figures, tables, appendix tables,
↪  catalogs, and reproducible scripts inside allowed write roots.
- Write catalog entries for every display artifact.
- A `qualitative` role must contain inspectable examples, not only
↪  aggregate scores.
- If a display role cannot be materialized, write an omission note in
↪  `display_materialization_report.json` instead of redefining the
↪  story.
- Do not replace the plan agent, change display roles, write section
↪  prose, or redefine the paper story.

Qualitative display example:
- Correct: a poem-reversal table with prompt, visible continuation,
↪  model output, and direction.
- Wrong: a table with only `Model`, `Forward`, and `Reversal` scores.
```

### E.4 Section Writing

Section writers distinguish facts available in the research materials from concepts already introduced to the reader. The instructions connect local section responsibilities to the paper's argument and discourage prose that only repeats a plan or lists numerical results.

**Source:** `model_section.md`

```
Read the current TaskSpec and any retry brief. Begin with the
↪  `Manuscript
context` block in the TaskSpec and read the listed reader-visible
↪  prior sections
to establish what the reader can already know. Then read
`paper/plans/story_plan.md` as the narrative authority and the current
`section_plan.json` entry as the local writing plan. Finally, read
↪  later-section
factual sources and any evidence, citation, display, appendix, or
↪  warning
artifacts needed for accuracy. Those sources control facts and
↪  required objects;
they do not supply manuscript voice or reader-visible prior knowledge.
This sequence refines any general instruction to read Story first:
↪  Story remains
the narrative authority, but establish the reader-visible boundary
↪  before using
it to direct the prose.

[...]

- Give the reader the answer early, after establishing the minimum
↪  concepts,
  comparisons, and facts needed to understand that answer.
- Complete the section job from the TaskSpec instructions and section
↪  plan.
- Use the TaskSpec's reader-visible section list to decide which
↪  terms,
  examples, facts, and conclusions may be treated as shared knowledge.
  ↪  As the
  current section develops, its completed sentences and paragraphs
  ↪  naturally
```

```
  join that reader-visible context.
- Use factual-only section inputs to make synthesis accurate, but do
↪  not assume
  that their definitions, wording, examples, or referents have already
  ↪  appeared
  in the manuscript. Reintroduce any information taken from a later
  ↪  section so
  that a reader encountering it here for the first time can understand
  ↪  it.

[...]

- Make the prose discovery-driven. Every paragraph must advance
↪  problem
  understanding, mechanism or discovery, method necessity, evidence
  ↪  payoff, or
  implication. The next paragraph must follow because the previous
  ↪  paragraph
  changed the reader's understanding, not because the section plan
  ↪  listed
  another field. Do not write process roadmaps such as `we develop the
  ↪  argument
  in N steps`, and do not turn required facts, citations, displays, or
  ↪  results
  into an equal-weight inventory.
```

### E.5 COLD-READER REVIEW

Cold-reader review first produces a natural-language response to the manuscript snapshot. A second prompt continues the same session and converts that response into a structured revision decision without adding new criticism. Major issues must require changes to the section under review; minor issues do not trigger a retry.

**Source:** `model_cold_reader_review.md`

```
You are reading a manuscript section for the first time as a research
↪  colleague.

Read only `{cold_reader_input_ref}` and follow the manuscript in the
↪  order shown
there. The snapshot identifies the current section type and whether
↪  the reading
context is a sequential prefix or the whole generated manuscript. Do
↪  not inspect
task plans, Story, reviewer memory, data, citation sources, display
↪  plans, skills,
or any other project file.

Use the orientation for the current section as a reading lens rather
↪  than a
fixed template, target length, or required sequence. A coherent
↪  section may
realize these roles in another order.

Respond as a thoughtful human reader in a continuous reading response.
↪  Judge whether
the current section feels like the kind of paper section it claims to
↪  be and
whether it works in the narrative available to you. Discuss what is
↪  clear and
worth preserving. Where the reading experience calls for it, explain
↪  narrative
```

```
breaks, weak motivation, distracting detail, defensive prose, claim
↪  scope,
transitions, or an ending that does not land naturally. These are
↪  examples of
useful reader reactions, not dimensions that must all be covered.

Do not force criticism. It is valid to conclude that the section
↪  already works.
When a concern means the section needs revision, say so directly;
↪  distinguish it
from optional or minor refinement without turning the response into a
↪  rubric.
Keep that judgment scoped to the current section. A question or
↪  obligation that
can be answered entirely by a later section is useful forward-looking
↪  feedback,
but it does not by itself mean the current section needs revision.
↪  Treat it as a
current-section revision only when the prose visible here also needs
↪  to change,
and explain that local change.
Do not fact-check against unavailable sources and do not rewrite the
↪  section.
Give complete natural-language feedback with enough context for the
↪  writer to
understand the reading experience.
```

**Source:** `model_cold_reader_projection.md`

```
Continue the same cold-reader session. Project the natural feedback in
`{cold_reader_feedback_ref}` into the ColdReaderReport at
`{cold_reader_review_report_ref}`.

Return the report as JSON with `summary`, `strengths_to_preserve`,
↪  `issues`,
`decision`, and `retry_guidance`. For every issue, directly choose the
↪  semantic
`severity` value `major` or `minor` in the JSON. Directly set
↪  `decision` to
`needs_revision` when at least one issue is `major`, and to `pass` when
↪  all issues
are `minor` (or there are no issues). Only explicit `major` issues
↪  trigger a
retry. Minor issues produce warnings only: preserve their
↪  natural-language
content in the report, but do not include their required changes in
`retry_guidance`.

Make severity task-scoped. An issue is `major` only when resolving it
↪  requires a
change to the current reviewed section. A conditional obligation owned
↪  entirely
by a later section is not a current-section `major`; preserve it as a
↪  `minor`
forward-looking warning unless the feedback also identifies prose in
↪  the current
section that must change.

This JSON is a faithful downstream representation of the original
↪  response. Do
not add a concern that the feedback did not raise or change its
↪  substance.
Preserve the strengths the response identified, and keep each issue's
↪  location,
```

```
message, and required change close to the original feedback so the
↪  projection
remains auditable. Do not add a `feedback_basis` field.
```

### E.6 SEMANTIC INTEGRATION

The semantic integration pass reads sections in manuscript order before consulting the planning and evidence files. It then repairs definitions, cross-section consistency, and claim scope, and synthesizes the abstract from the finalized body.

**Source:** `model_finalization_semantic.md`

```
Use staged disclosure inside this one invocation only:
1. Read the pass spec, then read only `paper_plan.section_order` to
↪  establish the
   canonical manuscript sequence. Do not open Story, section plans,
   ↪  evidence,
   warnings, or artifact indexes yet.
2. Cold-read every existing canonical section in that sequence. Start
↪  Abstract
   from empty reader context, then reset to empty context for
   ↪  Introduction; do
   not carry Abstract wording into the body reader's prior knowledge.
   ↪  Beginning
   with Related Work, or the next canonical body section when Related
   ↪  Work is
   absent, carry forward only information explicitly established by
   ↪  preceding
   body prose. Within each section, completed sentences and paragraphs
   ↪  may
   support what follows in that same section.
3. After the ordered cold-read, open Story, section and evidence
↪  plans, upstream
   warnings, citation and display artifacts, figure/table and appendix
   ↪  indexes,
   and raw evidence as needed for factual verification and scope.
4. Complete semantic repairs and Abstract synthesis. Information
↪  learned from a
   later section or factual source must be introduced where it first
   ↪  appears;
   plan fields and Story goals do not count as reader-visible setup.
   ↪  Worked
   examples must expose every condition used by their reasoning, or be
   ↪  omitted
   and reported when the supplied information is insufficient.
5. Reread the revised canonical sections in the same order with the
↪  same
   Abstract and Introduction resets. Repair any undefined term, hidden
   ↪  premise,
   unresolved referent, or later result treated as shared knowledge
   ↪  before
   writing semantic artifacts.

Semantic responsibilities:
- You must create or rewrite `paper/sections/abstract.tex` from the
↪  finalized body, evidence, limitations, and conclusions.
- Repair cross-section consistency, supported claim wording, citation
↪  alignment, terminology, transitions, appendix bridges, and
↪  control-plane leakage.
```

# F CORE WRITING SKILL EVOLUTION

Our evolution turns experience from completed writing tasks into reusable procedures within an existing harness and skill foundation. Each candidate targets one of Plan, Writing, or Finalize, holding the other two modules and their skills fixed. Harness evolution permits changes to the selected module and its associated skills. Once the harness is fixed at $H_m$, updates are restricted to the corresponding skills, with task coordination and tool use unchanged. We examine core instruction growth across historical checkpoints, which may aggregate multiple accepted updates and span harness changes.

**Core writing scope.** Checkpoint 0 represents the initial writing baseline; 1, the intermediate integrated writing system; 2, the earlier modular harness organized into Plan, Writing, and Finalize; and 3, the current system after further refinement. Token counts cover only core writing guidance, excluding auxiliary skills.

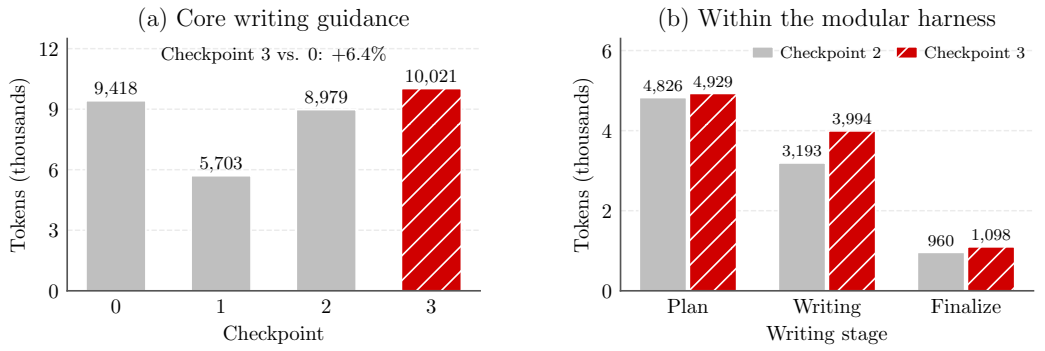



Figure 13: Core writing instruction size. (a) Total tokens in one integrated skill (checkpoints 0–1) or three stage-specific skills (2–3). (b) Per-stage counts for the modular harness. Red hatched bars denote the current checkpoint. Checkpoints are historical milestones, not individual optimization iterations.

**Further refinement without explosive core growth.** Figure 13(a) shows that the current three-skill core uses 6.4% more tokens than the initial integrated skill. The shorter intermediate skill shows that development includes compression as well as expansion. Within the modular harness, the same three core skills grow by 11.6% from checkpoint 2 to 3. These observed checkpoints show no explosive growth in core instruction length or proliferation of core skill entrypoints between the modular checkpoints. This pattern is consistent with concentrating experience into reusable guidance within existing roles.

**From writing diagnoses to reusable procedures.** Figure 13(b) locates approximately 77% of the net token increase in Writing. One edit strengthens "read preceding sections" into a reader-knowledge boundary: later inputs may supply facts, but only preceding manuscript text establishes what the reader already knows. This addresses passages that are factually correct yet assume definitions or motivations introduced later. One principle governs concept introduction, argument order, and transitions within the existing writing role. It illustrates the abstraction sought by our evolution loop: Gap Analysis removes reference-only facts and copyable wording, Wiki retains reusable lessons, and Skill Proposal turns permitted lessons into bounded skill edits. The persistent change is a writing procedure applicable to new materials. Under the fixed harness, paired validation compares original and candidate skills on the same new materials, assessing complete manuscripts for quality, readiness, and lower-tail performance. Only approved skills become active for future papers, linking refinement to performance beyond the triggering paper. The observed compact core supports this route to improving an existing writing system through reusable procedures.