Skills, Benchmarks, and Verification Are What AI-Assisted Research Needs

Patrik Reizinger & Wieland Brendel
Max-Planck-Institute for Intelligent Systems, ELLIS Institute Tübingen, University of Tübingen
`{patrik.reizinger, wieland.brendel}@tuebingen.mpg.de`

Abstract

AI adoption in research has outpaced our understanding of its limitations. Researchers embraced these tools for everything from literature review to code generation, yet the *jagged frontier*—the uneven boundary where AI excels at some tasks and fails unpredictably at others—remains unmapped. We already see real-world consequences: papers with fabricated citations passed peer review at NeurIPS 2025, and the rising tide of "AI slop"—low-quality, machine-generated submissions—is straining review systems across venues. A preliminary survey of AI researchers reveals a similar finding: while many actively use vanilla AI coding pipelines, there is relatively little adoption and development of robust and verified workflows for other parts of the scientific process like ideation, literature research, or experiment design. This suggests the path forward: build infrastructure for *discovery*, *comparison*, and *verification*. We propose the *Research Agora* to close this gap—a *marketplace* for discovering reusable AI workflows (skills), *benchmarks* for comparing their effectiveness, and *test-driven research* for verifying outputs before they propagate. We have built working examples—from reference checking that catches hallucinated citations to structured writing with layered quality checks—demonstrating how different verification levels apply to different tasks. We release these as a starting point and call on the research community to extend, improve, and benchmark them.[1]

1 Introduction

Field experiments reveal a "jagged technological frontier"—an uneven boundary where some tasks are easily accomplished by AI while others, though seemingly similar in difficulty, lie outside its current capabilities (Dell'Acqua et al., 2023). AI excels at code autocompletion but hallucinates citations in similar-looking text generation tasks. GPTZero analyzed nearly all papers accepted to NeurIPS 2025 and found multiple hallucinated citations (GPTZero, 2026), findings subsequently validated by independent analysis (Ansari, 2026). If these had been feature requests for software, automated checks would have caught errors. This exposes a critical gap: **most researchers are not operating at the frontier of what AI tools can do** (Wang et al., 2023), yet we lack the verification infrastructure to safely explore it. Software engineers have mapped this frontier through distributed experimentation; research faces unique verification challenges. Our preliminary survey of 38 researchers (Section A; note methodological caveats therein) suggests this tension: most use AI daily for research workflows (writing, literature review, automation), yet over two-thirds encounter errors weekly. These errors reflect a trifecta of problems: *AI slop* (low-quality, machine-generated submissions), *hallucinations* (e.g., citations), and *silent failures* (unlike code that crashes, research errors can go undetected for years).

Our survey and the NeurIPS incident reveal three intertwined barriers that we encountered firsthand while building AI-assisted workflows for citation verification, scientific writing, and experiment planning:

1. **Discovery problem:** Over a third do not know what AI tools exist for research; we built skills others might need, while others have surely built skills we could use—but no infrastructure connects us.

[1] Skills marketplace: `https://rpatrik96.github.io/research-agora/`

2. **Comparison problem:** Without shared benchmarks, we cannot know if our citation verification approach is better or worse than alternatives; the field relies on anecdotes and popularity metrics.
3. **Verification/Trust problem:** Most researchers verify outputs manually, but the NeurIPS incident shows errors propagate silently at scale—unlike software where compilers and tests can catch mistakes immediately.

These barriers make adoption harder than it needs to be. Our survey reveals a striking pattern: researchers verify AI-generated code most of the time ("Essential/Often") versus relatively few for ideation (Section A). What explains the verification asymmetry? *The difference correlates with tooling availability.* Software engineering demonstrates this pattern: package managers enable discovery (npm, pip), benchmarks enable comparison (SWE-bench (Jimenez et al., 2024), HumanEval (Chen et al., 2021)), and CI/CD enables verification (pytest, GitHub Actions). Research lacks this *integrated* scaffold—fragmented tools exist, but the community infrastructure connecting them does not.

We propose the **Research Agora**—an integrated ecosystem addressing all three problems (Section 2). It combines a skills marketplace for discovery, benchmarks for comparison, and **Test-Driven Research (TDR)** for verification. Each component alone is insufficient; together they enable researchers to collectively map the jagged frontier (Section 2.2). But TDR addresses the "how," not the "what": as execution becomes delegable, **taste**—the judgment to identify meaningful problems—becomes irreplaceable, yet develops through deliberate practice (Section 3). We release the working infrastructure for two pillars—a skills marketplace with 74 workflows across 6 plugins and TDR verification tools—but also provide a blueprint for benchmarks, e.g., for citation verification. However, we consider what we built only as a starting point; we need a community effort to develop a joint expertise for accelerating research with verified AI assistance (Section 2).

2 The Research Agora: A Research Agenda

Software engineers mapped their jagged frontier through distributed experimentation—sharing tools, comparing performance, and building verification infrastructure. Research lacks this ecosystem. We propose the **Research Agora**—working infrastructure we built for our own research, released as a starting point and a *call to action* for the community to extend.

2.1 Three Pillars: Discovery, Comparison, Verification

The goal is to *lower the barrier to confident adoption of AI-assistance for research*—reducing friction through three integrated components. A **skills marketplace** enables discovery, drawing on the open-source ethos of community-driven contribution (Raymond, 1999): researchers share *skills*—modular AI workflows encoded as structured prompts with task definitions, verification criteria, and potential failure modes. **Benchmarks** enable comparison: aggregated test suites providing quantitative assessment ("catches 92% of fabricated citations") rather than anecdotes. **Test-Driven Research (TDR)** enables verification: defining acceptance criteria *before* delegating to AI (Section 2.3).

Each alone is insufficient: visibility does not imply quality, benchmarks risk Goodhart's law, and verification without comparison cannot rank alternatives. The Agora lives at their intersection (Figure 1). This vision is materializing in software—AgentSkills.io (Anthropic, 2025a), Tessl (Tessl, 2026), Claude Code (Anthropic, 2025b)—but research needs layered verification suited to prose and claims, not just executable code.

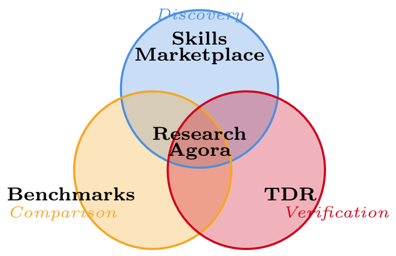

Figure 1: The Research Agora: three pillars, each insufficient alone.

Mapping what is possible: our AI research workflow. Following the principle "build what you need and use what you build" (Black, 2024), we release working infrastructure and propose a research agenda for the three pillars: **(1) Discovery:** A skills marketplace with 74 modular workflows across 6 plugins spanning the research lifecycle—writing, verification, dissemination, and specialized agents (Section D).

(2) Verification: TDR workflows including citation verification (catching NeurIPS-style hallucinations via DOI resolution) and code-paper consistency checks. **(3) Comparison:** As a first step, we plan to release a citation hallucination benchmark with hidden test sets for evaluating detection methods[2]. The Agora requires all three; we contribute the vision, working examples, and a concrete agenda for what remains.

2.2 The Verification Gap

Table 1 quantifies the asymmetry: *trust correlates with tooling.* Our survey shows coding adoption far exceeds ideation—researchers delegate confidently where verification exists (Section A). This suggests: build verification infrastructure, and adoption follows (though not every research task can be verified).

Aspect	**Software Engineering**	**Research**
Verification	Compilers, tests, CI/CD	Peer review, replication
Failure mode	Fails loudly (error, crash)	Fails silently
Feedback loop	Immediate (seconds)	Delayed (months/years)
Shared standards	Specs, linting, types	Few; varies by subfield

Table 1: Verification infrastructure comparison. Software catches errors immediately; research errors propagate silently. This analogy has limits: software correctness is more formalizable than research significance.

2.3 Verification in Practice: Test-Driven Research

We focus on verification because it is the least developed pillar—discovery has partial analogues (citation managers, tool directories; benchmarks are wide-spread in deep learning), but systematic verification infrastructure for research does not exist.

The Core Principle. Software engineering's test-driven development writes tests *before* implementation (Beck, 2003). We propose **Test-Driven Research (TDR)**: define verification criteria *before* delegating tasks to AI.

Verification exists on a spectrum trading certainty for scope (Figure 2) (Mayo, 2018). *Formal tests* (DOI resolution, code execution) provide high certainty on narrow claims; *peer review* evaluates broad significance but sacrifices precision (National Academies of Sciences, Engineering, and Medicine, 2019). Match verification strength to claim type: use automated tests for checkable facts, peer review for novelty judgments.

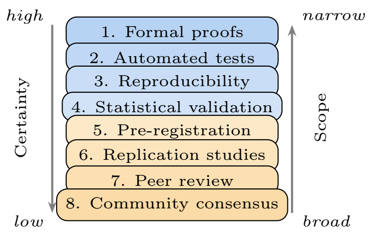

Figure 2: TDR verification hierarchy: certainty vs. scope tradeoff.

From Unit Tests to Benchmarks. Individual TDR test cases function as unit tests for a skill; aggregated test suites become benchmarks for comparing skills. A reference verification skill might include unit tests verifying DOI resolution, title matching, and author list consistency. Aggregate these across a diverse bibliography, and you have a benchmark enabling objective comparison: "Skill A catches 92% of hallucinations vs. Skill B's 78%." Concretely, such a benchmark partitions entries into public development and hidden test sets updated quarterly—the temporal segmentation that makes LiveCodeBench robust (Jain et al., 2024). Recent work confirms the urgency: Xu et al. (2026) find that 1.07% of papers across 56K top-tier publications contain invalid citations (80.9% increase in 2025), while Sakai et al. (2026) identify nearly 300 affected papers in ACL venues alone. Open-source tools such as HaRC (HaRC Contributors, 2024), CiteVerifier (Xu et al., 2026), hallucinator (Barbosa, 2024), and verify-citations (verify-citations Contributors, 2026) already verify citations against scholarly databases—a standardized benchmark would enable objective comparison of these approaches. Since submission, we have developed HALLMARK—a citation hallucination

[2]Sicne submitting the paper, we implemented the benchmark. Code: `https://github.com/rpatrik96/hallmark`

benchmark with 2,525 annotated entries across 14 hallucination types and built-in baselines—demonstrating the benchmarks pillar is actionable. We plan to release it alongside additional benchmarks for other research skills.

Coding benchmarks demonstrate this progression: HumanEval (Chen et al., 2021), SWE-bench (Jimenez et al., 2024), and LiveCodeBench (Jain et al., 2024) aggregate unit tests into benchmarks with hidden test sets and temporal segmentation. These design principles transfer to research: define acceptance criteria, use hidden test sets, track temporal provenance.

Different research tasks afford different verification levels. Citation validation enables formal verification with clear pass/fail criteria; scientific writing admits *layered* verification—style rules (passive voice, filler phrases, sentence length) can be checked automatically, while audience alignment and argument flow require heuristic LLM-assisted judgment. TDR's power comes from making criteria explicit *before* delegation—also acknowledging that the extent of which, and even the possibility varies substantially across tasks—; the Agora provides infrastructure for encoding these standards into shareable skills.

TDR verifies correctness, not importance—you can verify the wrong hypothesis perfectly. For checkable facts (citations exist, code executes, statistics are valid), TDR provides strong guarantees. For creative contributions (which problems matter, which approaches are promising), taste remains irreducibly human. TDR also has failure modes: poorly specified criteria produce false confidence, and verification itself may require the expertise being delegated.

Where TDR is hard to apply. TDR's applicability varies: formal verification (DOI resolution, code execution) works well for checkable facts, and heuristic checks (writing quality, statistical validation) provide useful but imperfect signals. However, some contributions resist verification entirely: theoretical work requires taste to evaluate proof elegance; qualitative research depends on interpretive judgment; novel experimental designs demand the very expertise TDR aims to verify. The verification hierarchy is best understood as a spectrum—match verification strength to what each task affords, rather than expecting uniform coverage.

As execution becomes delegable, *taste*—the judgment to identify problems worth solving (Graham, 2004; Hamming, 1986)—becomes the bottleneck.[3]

We illustrate TDR concretely through three worked examples—citation verification, code-paper consistency, and structured writing with layered checks—in Section C.

3 Discussion and Conclusion

The Research Agora addresses three barriers to AI adoption: discovery (marketplace), comparison (benchmarks), and verification (TDR). Their integration provides institutional infrastructure research currently lacks.

Limitations. Potential downsides include deskilling (Newport, 2025), homogenization (Yakura et al., 2024), and reduced methodological diversity. Research taste (Graham, 2004; Hamming, 1986) develops through struggle, yet delegation may bypass this—our survey confirms "reduced learning for students" ranks second among concerns (Sections A and B). These risks motivate the Agora's verification infrastructure: navigate adoption deliberately, not blindly.

Open Questions. Key questions remain: *Governance*—who curates the marketplace and prevents capture? *Incentives*—can we create citation-like credit for shared workflows? *Scope*—which research tasks resist formalization entirely? *Pedagogy*—how should PhD programs balance delegation with deliberate practice?

Call to Action. We release working skills as a starting point.[4] Individual researchers can adopt TDR today; institutions can integrate verification into review pipelines. The Research Agora requires collective effort: by jointly building verified AI-tooling for research, the whole community benefits.

[3] Concurrent work demonstrates TDR principles: Woodruff et al. (2026) report "neuro-symbolic" loops where AI writes and executes code to verify derivations.

[4] `https://rpatrik96.github.io/research-agora/`

Acknowledgements

Patrik Reizinger acknowledges his membership in the European Laboratory for Learning and Intelligent Systems (ELLIS) PhD program and thanks the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for its support. This work was supported by the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039A. Wieland Brendel acknowledges financial support via an Emmy Noether Grant funded by the German Research Foundation (DFG) under grant no. BR 6382/1-1 and via the Open Philantropy Foundation funded by the Good Ventures Foundation. Wieland Brendel is a member of the Machine Learning Cluster of Excellence, EXC number 2064/1 – Project number 390727645. This research utilized compute resources at the Tübingen Machine Learning Cloud, DFG FKZ INST 37/1057-1 FUGG.

References

Samar Ansari. Compound deception in elite peer review: A failure mode taxonomy of 100 fabricated citations at NeurIPS 2025. *arXiv preprint arXiv:2602.05930*, 2026.

Anthropic. Agent skills: An open standard for ai agent capabilities. `https://agentskills.io/`, 2025a.

Anthropic. Claude code: Plugins and marketplaces. `https://docs.anthropic.com/en/docs/claude-code/plugins`, 2025b.

Gianluca S. Barbosa. hallucinator: Citation verification against scholarly databases, 2024. URL `https://github.com/gianlucasb/hallucinator`.

Kent Beck. *Test-Driven Development: By Example*. Addison-Wesley, Boston, 2003.

Michael J. Black. Build what you need and use what you build. `https://perceiving-systems.blog/en/post/build-what-you-need-and-use-what-you-build`, 2024.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chanez, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Fabrizio Dell'Acqua, Edward McFowland, Ethan R. Mollick, Hila Lifshitz-Assaf, Katherine Kellogg, Saran Rajendran, Lisa Krayer, François Candelon, and Karim R. Lakhani. Navigating the jagged technological frontier: Field experimental evidence of the effects of AI on knowledge worker productivity and quality. *Harvard Business School Working Paper*, (24-013), 2023.

Tyna Eloundou, Sam Manning, Pamela Mishkin, and Daniel Rock. GPTs are GPTs: An early look at the labor market impact potential of large language models. *Science*, 2024. doi: 10.1126/science.adj0998.

GPTZero. GPTZero finds 100 new hallucinations in NeurIPS 2025 accepted papers. `https://gptzero.me/news/neurips/`, 2026.

Paul Graham. Taste for makers. `http://www.paulgraham.com/taste.html`, 2004.

Richard W. Hamming. You and your research. `https://www.cs.virginia.edu/~robins/YouAndYourResearch.html`, 1986.

HaRC Contributors. HaRC: Hallucinated reference checker, 2024. URL `https://pypi.org/project/harcx/`.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2024.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024.

Kevin Kelly. Kevin kelly points a new way forward into the ai age. `https://peterleyden.substack.com/p/kevin-kelly-points-a-new-way-forward`, 2025.

Deborah G. Mayo. *Statistical Inference as Severe Testing: How to Get Beyond the Statistics Wars*. Cambridge University Press, Cambridge, 2018. doi: 10.1017/9781107286184.

Ethan Mollick. The cybernetic teammate. `https://www.oneusefulthing.org/p/the-cybernetic-teammate`, 2025a.

Ethan Mollick. Management as AI superpower. `https://www.oneusefulthing.org/p/management-as-ai-superpower`, 2025b.

National Academies of Sciences, Engineering, and Medicine. *Reproducibility and Replicability in Science*. The National Academies Press, Washington, DC, 2019. doi: 10.17226/25303.

Cal Newport. *So Good They Can't Ignore You: Why Skills Trump Passion in the Quest for Work You Love*. Grand Central Publishing, New York, 2012.

Cal Newport. *Deep Work: Rules for Focused Success in a Distracted World*. Grand Central Publishing, New York, 2016.

Cal Newport. Be wary of digital deskilling. `https://calnewport.com/be-wary-of-digital-deskilling/`, 2025.

Shane Parrish. The surprising reason writing remains essential in an AI-driven world. `https://fs.blog/why-write/`, 2024.

Eric S. Raymond. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly Media, Sebastopol, CA, 1999.

Yusuke Sakai, Hidetaka Kamigaito, and Taro Watanabe. HalluCitation matters: Revealing the impact of hallucinated references with 300 hallucinated papers in ACL conferences. *arXiv preprint arXiv:2601.18724*, 2026.

Judy Hanwen Shen and Alex Tamkin. How AI impacts skill formation. *arXiv preprint arXiv:2601.20245*, 2026.

William Strunk, Jr. and E. B. White. *The Elements of Style*. Longman, New York, 4th edition, 2000.

Tessl. Tessl: The package manager for agent skills. `https://tessl.io/`, 2026.

verify-citations Contributors. verify-citations: Automated citation verification tool, 2026. URL `https://github.com/vishakhpk/verify_citations`.

Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, Joseph Anber, Ryan Yasonik, et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620:47–60, 2023. doi: 10.1038/s41586-023-06221-2.

David P. Woodruff, Vincent Cohen-Addad, Lalit Jain, Jieming Mao, Song Zuo, Mohammad-Hossein Bateni, Simina Branzei, Michael P. Brenner, Lin Chen, Ying Feng, Lance Fortnow, Gang Fu, Ziyi Guan, Zahra Hadizadeh, Mohammad T. Hajiaghayi, Mahdi JafariRaviz, Adel Javanmard, C. S. Karthik, Ken-ichi Kawarabayashi, Ravi Kumar, Silvio Lattanzi, Euiwoong Lee, Yi Li, Ioannis Panageas, Dimitris Paparas, Benjamin Przybocki, Bernardo Subercaseaux, Ola Svensson, Shayan Taherijam, Xuan Wu, Eylon Yogev, Morteza Zadimoghaddam, Samson Zhou, and Vahab Mirrokni. Accelerating scientific research with Gemini: Case studies and common techniques. *arXiv preprint arXiv:2602.03837*, 2026.

Zuyao Xu, Yuqi Qiu, Lu Sun, FaSheng Miao, Fubin Wu, Xinyi Wang, Xiang Li, Haozhe Lu, ZhengZe Zhang, Yuxin Hu, Jialu Liu, Jin Luo, Feng Zhang, Rui Luo, Xinran Liu, Yingxian Li, and Jiaji Liu. GhostCite: A large-scale analysis of citation validity in the age of large language models. *arXiv preprint arXiv:2602.06718*, 2026.

Hiromu Yakura, Ezequiel Lopez-Lopez, Levin Brinkmann, Ignacio Serna, Prateek Gupta, Ivan Soraperra, and Iyad Rahwan. Empirical evidence of large language model's influence on human spoken communication. *arXiv preprint arXiv:2409.01754*, 2024.

A Survey: AI Tool Usage in Research

To ground our arguments empirically, we conducted a preliminary survey of 38 researchers on AI tool usage patterns, verification practices, and concerns. **Note:** This survey is exploratory and not representative—the convenience sample is small and self-selected, limiting generalizability. We present these results as illustrative evidence rather than definitive findings. A larger, stratified survey is planned as future work.

Methodology. We distributed an anonymous online questionnaire (15 questions) via research lab networks and social media (X/Twitter) over two weeks in late 2025; participation was voluntary with no compensation. The sample is predominantly ML/AI researchers (84%), spanning PhD students (47%), postdocs (18%), research scientists (16%), undergrad/master's (11%), and faculty (5%). Given the small sample size and convenience sampling, all findings should be interpreted as preliminary observations motivating our research agenda rather than generalizable claims.

A.1 Adoption Patterns

Usage frequency. AI tool adoption is high: two-thirds use AI tools daily, with relatively few using them weekly or less. ChatGPT leads adoption (100%), followed by Claude (most respondents) and GitHub Copilot/Cursor (two-thirds). This suggests AI assistance has become integral to research workflows.

Task-specific adoption. Table 2 reveals a striking pattern: **coding dominates** (a strong majority use AI "Essential" or "Often"), while tasks lacking verification infrastructure see lower adoption. Writing shows moderate adoption (over a third "Essential/Often"), while ideation (relatively few) lags behind—precisely those tasks where outputs are hardest to verify.

Table 2: AI tool usage frequency by research task (n=38). Tasks with better verification infrastructure (coding) show higher adoption than tasks resisting verification (ideation).

Task	**Essential**	**Often**	**Sometimes**	**Rarely**	**Never**
Coding & implementation	29%	45%	16%	3%	8%
Writing (drafts, LaTeX)	8%	29%	32%	21%	11%
Figures & visualizations	26%	26%	18%	11%	18%
Data analysis	21%	16%	21%	11%	32%
Literature review	11%	24%	26%	29%	11%
Ideation/hypotheses	3%	8%	26%	37%	26%
Feedback/simulated review	11%	18%	16%	26%	29%
Administrative tasks	5%	16%	16%	26%	37%

When asked which task benefits *most* from AI, over half selected coding—the task with the strongest verification infrastructure (tests, compilers, execution). This supports our argument that verification enables confident delegation.

A.2 The Verification Challenge

Error frequency. Researchers encounter AI errors frequently: over two-thirds report hallucinations or errors weekly or more, while about a third encounter them monthly. This highlights the practical importance of verification.

Verification strategies. Respondents employ multiple simultaneous verification methods (Table 3). Manual review dominates (nearly all respondents), but notably, "run and test code" (a strong majority) ranks second—again highlighting verification infrastructure's importance. Only a small minority generally trust outputs without verification, suggesting researchers are aware of reliability limitations.

Table 3: Verification methods used for AI outputs (multiple selections allowed, n=38). Researchers employ multiple strategies, with code testing second only to manual review.

Verification Method	% Using
Manual review of outputs	95%
Run and test generated code	74%
Domain expertise/intuition	53%
Cross-reference other sources	50%
Only use for low-stakes tasks	39%
Use another AI to check	13%
Generally trust without verification	5%

A.3 Barriers and the Case for a Marketplace

Discovery problem. Barriers to adoption support the marketplace concept: over a third don't know what tools exist, nearly half find benefits unclear for their work, and half cite cost. When asked about interest in a "curated directory of research-specific AI tools and workflows," respondents showed moderate to strong interest (mean: 3.68 out of 5), suggesting demand for organized discovery.

Discovery channels. Researchers currently learn about tools through colleagues (about half), social media (about half), and self-exploration (nearly half). Only a small minority cite conferences. A structured marketplace could systematize this ad-hoc discovery.

A.4 Concerns: The Generational Challenge

Top concerns. Table 4 reveals that beyond output quality (cited by most researchers), **"reduced learning for students" ranks second (also cited by most)**—directly supporting our generational challenge argument (Section 3). Researchers are already worried about taste development in the next generation.

Table 4: Top concerns about AI in research (up to 3 selections, n=38). Reduced learning for students ranks second, validating the generational challenge argument.

Concern	% Selecting
Quality/reliability of outputs	82%
Reduced learning for students	79%
Dependence on commercial providers	61%
Homogenization of research	42%
Widening resource gap between labs	34%
Unclear attribution/credit	26%
Privacy of research data	26%

Productivity expectations. Researchers expect substantial impact: over half anticipate 25%+ productivity gains within 2-3 years, while relatively few expect minor or no impact. Combined with reliability concerns, this suggests researchers see high potential but recognize verification as a bottleneck.

Disclosure practices. Among those who have published with AI assistance (n=25), few always disclose, roughly half sometimes disclose (substantial contributions only), and the remainder rarely or never disclose. The lack of consensus reflects broader uncertainty about norms—another area where community deliberation through an agora could help.

B Taste and Training the Next Generation

This appendix expands on the reverberations of AI-assisted research for taste development and PhD training, which were compressed in Section 3 to maintain focus on the Research Agora proposal.

B.1 Taste: The Unmeasurable Human Contribution (for Now)

AI shifts the researcher's role from executor to orchestrator (Eloundou et al., 2024; Mollick, 2025a). Management becomes an "AI superpower" (Mollick, 2025b): the bottleneck is no longer execution but *deciding what to build*. This raises a fundamental question: *Can you manage what you've never done?*

Following Graham (Graham, 2004) and Hamming (Hamming, 1986), *taste* means recognizing important problems and viable approaches. Hamming: "It's not the consequence that makes a problem important, it is that you have a reasonable attack." AI can execute but cannot have taste—LLMs write papers but cannot identify which are worth writing. Whether this is temporary or principled remains contested; we take it as a working hypothesis that taste requires situated experience AI currently lacks. Taste resists formalization: any criterion for evaluating taste itself requires taste to apply (the regress problem).

Taste develops through struggle (Kelly, 2025; Newport, 2012). Working at the edge of your abilities builds not just skill but *judgment*. Crucially, *writing is thinking* (Parrish, 2024): the struggle to articulate reveals gaps; structuring forces synthesis. Be strategic: delegate routine tasks, but core intellectual work requires the struggle that builds understanding.

B.2 Implications for PhD Training

Senior researchers developed taste before AI; they delegate confidently because they know what good work looks like. PhD students today may never engage deeply with foundational tasks—and taste has no formal verifier. Our preliminary survey indicates: "reduced learning for students" ranks second among concerns (cited by most respondents), behind only output quality (Section A).

Emerging evidence supports this: developers using AI showed impaired conceptual understanding despite productivity gains—though certain engagement patterns preserved learning, suggesting the problem is *how* AI is used (Shen & Tamkin, 2026; Newport, 2025). If management becomes the "AI superpower," PhD programs must teach delegation and quality assessment—traditionally PI-level competencies—yet these presuppose knowing what good work looks like. The transition from technical execution to judgment, traditionally spanning PhD to PI years, is now compressed into the PhD itself. The goal: use AI strategically, building both taste from struggle and delegation skills (Newport, 2016).

The challenge is navigating this transition deliberately. Researchers must develop metacognitive awareness: which tasks build domain expertise versus which are safely delegable? The Research Agora can help by making these distinctions explicit in skill definitions—flagging which workflows preserve learning opportunities versus which purely optimize execution. But ultimately, taste cultivation requires individual commitment to deliberate practice on challenging problems, not just efficient delegation.

C TDR in Practice: Detailed Workflows

This appendix illustrates how Test-Driven Research principles can be applied to common research workflows. These examples are meant to inspire further thought rather than prescribe specific implementations—the key insight is defining verification criteria *before* delegating tasks to AI, regardless of the particular tools used.

C.1 Literature Review: Automated Citation Verification

The challenge. AI assistants frequently hallucinate citations, as demonstrated by the NeurIPS 2025 incident. Traditional manual verification is tedious and error-prone at scale.

TDR approach. Define "valid citation" formally *before* delegating literature review to AI: every BibTeX entry must have a DOI or arXiv ID that resolves correctly, titles must match database records (CrossRef, Semantic Scholar, DBLP) within some edit distance threshold, and author lists must be consistent with authoritative sources.

Tools can then automatically validate bibliographies by querying scholarly databases and flagging entries that fail these criteria.[5]. Researchers inspect flagged citations manually to determine whether the AI hallucinated papers, confused similar titles, or made legitimate errors.

Why this embodies TDR. Citation validity is transformed from subjective human judgment to automated verification. The NeurIPS incident would have been prevented had conferences required automated citation checking in their submission pipeline. Importantly, this doesn't eliminate human judgment—it focuses human attention on genuine ambiguities rather than mechanical checking (Figure 3).

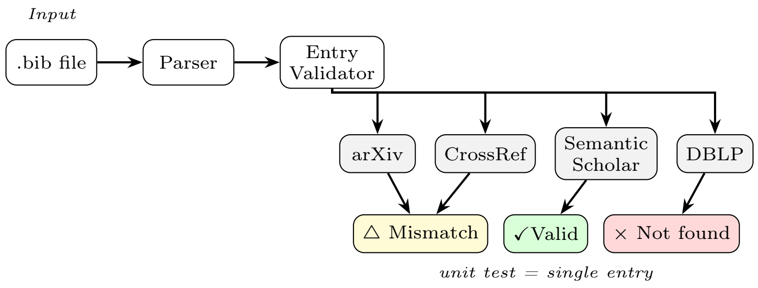

Figure 3: BibTeX verification pipeline. Each entry is validated against scholarly databases; aggregated results form a benchmark for comparing citation-checking skills. Hidden test sets prevent gaming.

C.2 Code-Paper Consistency: Linking Implementations to Equations

The challenge. Research papers often describe algorithms mathematically while implementations differ subtly (or substantially) from the written equations. Common errors include: equations with typos that wouldn't compile, algorithms described one way but implemented differently, and mathematical claims unsupported by the actual codebase.

TDR approach. Establish bidirectional traceability between LaTeX equations and code implementations. Before drafting the paper, define acceptance criteria: every numbered equation describing an algorithm or formula must have a corresponding tested implementation, and every algorithmic implementation must reference its paper equation.

Tools can use annotations (e.g., Python decorators or structured comments) to link functions to equation labels, specifying test cases that define expected behavior. Automated verification checks that all equation labels referenced in code exist in the paper, all algorithmic equations have corresponding implementations, and all specified test cases pass.

Why this embodies TDR. Claims about algorithmic behavior become empirically testable. The verification criterion ("code matches equation and passes tests") is explicit and can be integrated into continuous integration pipelines. This preserves mathematical rigor while ensuring implementation fidelity.

C.3 Structured Writing: Layered Verification for Prose

The challenge. When AI generates prose, researchers risk losing narrative coherence and the "writing is thinking" benefit (Parrish, 2024). Unlike code, where compilers catch errors

[5]We release our solution at `https://github.com/rpatrik96/bibtexupdater`

immediately, prose quality spans a spectrum from mechanically checkable rules to irreducibly subjective judgments.

TDR approach. Apply *layered* verification that matches checking strength to claim type. Drawing on qualitative principles from Strunk & White's *Elements of Style* (Strunk & White, 2000), we define quantifiable automated checks: passive voice usage below 20%, elimination of filler phrases ("the fact that," "there is/are," "in terms of"), sentence length variance within bounds (avoiding both monotonous uniformity and runaway complexity), and adverb density checks. These rules are concrete enough to express as pass/fail tests—a prose analogue to unit tests for code.

The second layer requires heuristic, LLM-assisted judgment: Does the argument flow logically between paragraphs? Is technical depth calibrated to the target audience? Do transitions connect ideas rather than listing them? These checks cannot be reduced to simple metrics, but structured prompts (e.g., "identify paragraphs where the logical connection to the previous paragraph is unclear") make them systematic rather than ad-hoc.

The third layer remains human: Does this section advance the paper's thesis? Is the framing original or derivative? Does the narrative earn the reader's attention? No automated check substitutes for this judgment.

Why this embodies TDR. The layered structure defines verification criteria *before* delegation—the researcher specifies which style rules to enforce, which structural properties to check, and which aspects require personal review. This preserves intellectual ownership: the researcher controls argument structure and narrative choices while AI handles prose execution within explicit constraints. Crucially, the automated layers *focus* human attention on genuine intellectual challenges (argument quality, originality) rather than mechanical checking (passive voice counts, filler phrases).

Broader implications. These three examples—citation verification, code-paper consistency, structured writing—share a common pattern: formalize acceptance criteria before delegation, automate verification where possible, and focus human judgment on genuine intellectual challenges rather than mechanical checking. The specific tools matter less than the principle: test-driven workflows let researchers maintain control over "what" questions (which claims to make, which problems to solve) while safely delegating "how" questions (prose generation, mechanical checking) to AI.

D Current Implementation: Skills We Built

This appendix details the 74 skills we developed while conducting our own research, organized by category. These are released as a starting point—not as optimal solutions, but as working infrastructure others can use, extend, or replace.[6]

D.1 Academic Workflows (16 Skills)

Skills covering the full paper lifecycle:

Writing. `paper-introduction`, `paper-abstract`, `paper-experiments`, `paper-discussion`, `paper-summarizer`—each skill encodes section-specific guidance (what claims to make, what evidence to include, common pitfalls) while preserving researcher control over intellectual content.

Verification. `paper-references` validates BibTeX entries against CrossRef, Semantic Scholar, DBLP, and arXiv—catching the kind of hallucinated citations found in the NeurIPS incident. `paper-verify-experiments` checks code-paper consistency by linking implementations to equation labels.

Review. `paper-review` simulates skeptical reviewer feedback; `review-triage` categorizes and prioritizes reviewer comments for structured responses.

Dissemination. `paper-poster`, `paper-slides`, `science-gif`, `openreview-submission`—convert papers into venue-appropriate formats and handle submission workflows.

[6]Available at `https://github.com/rpatrik96/research-agora`.

Research Support. `literature-synthesizer` discovers and synthesizes relevant literature; `experiment-tracker` syncs experiment results to paper drafts; `benchmark-scout` identifies relevant benchmarks and generates experiment plans.

D.2 Formatting (2 Skills)

Publication-ready output: `latex-consistency` (enforce venue style guidelines), `tikz-figures` (neural network diagrams, flowcharts).

D.3 Research Agents (22 Agents, 12 Micro-skills, 4 Orchestrators, 3 Helpers)

Unlike skills (single-turn workflows), agents maintain context across interactions for deeper analysis. The plugin also provides composable micro-skills (atomic building blocks), orchestrators (multi-agent pipelines), and helpers (shared utilities).

Agents (22).

- **Adversarial analysis:** `devils-advocate` challenges arguments and identifies logical fallacies; `perspective-synthesizer` reconciles conflicting viewpoints into unified frameworks.
- **Audience and clarity:** `audience-checker` evaluates papers from different reader personas (reviewer, student, expert); `reader-simulation` simulates first-time reader comprehension; `voice-drift-detector` detects style inconsistencies across sections; `content-archaeologist` maps content structure for reorganization; `redundancy-radar` finds semantic overlap across documents.
- **Verification:** `claim-auditor` systematically traces all claims to evidence; `statistical-validator` checks p-values, confidence intervals, significance tests; `notation-consistency-checker` builds symbol tables and detects notation inconsistencies; `state-generator` creates structured paper representations for downstream analysis.
- **Theoretical analysis:** `proof-auditor` decomposes and verifies proofs step-by-step; `bounds-analyst` analyzes convergence rates and complexity bounds; `proof-strategy-advisor` suggests proof approaches; `counterexample-searcher` stress-tests theorems; `intuition-formalizer` translates informal intuitions into formal statements; `theory-connector` finds cross-domain connections; `theorem-dependency-mapper` builds dependency DAGs.
- **Publication:** `figure-storyteller` creates narrative-focused visualizations; `latex-debugger` diagnoses compilation errors; `artifact-packager` prepares code/data for release; `reviewer-response-generator` drafts rebuttals with evidence mapping.

Micro-skills (12). Atomic, composable building blocks that agents orchestrate: `claim-extractor`, `claim-classifier`, `evidence-locator`, `evidence-grader`, `citation-verifier`, `cross-referencer`, `assumption-analyzer`, `assumption-surfacer`, `novelty-checker`, `derivation-checker`, `proof-step-extractor`, `proof-step-verifier`.

Orchestrators (4). Multi-agent pipelines for comprehensive analysis: `parallel-audit` (concurrent claim verification), `parallel-review` (multi-perspective review), `parallel-theory-audit` (concurrent theoretical analysis), `pre-submission-audit` (end-to-end submission readiness check).

Helpers (3). Shared utilities: `batch-arxiv` (bulk arXiv retrieval), `context-compactor` (context window management), `prefetch-evidence` (evidence pre-loading for agents).

D.4 Development Automation (8 Skills)

Git workflow (`commit`, `pr-automation`), code quality (`code-simplify`, `python-cicd`), cluster computing (`htcondor`), and code-paper synchronization (`latex-sync-setup`, `latex-sync-annotate`, `latex-sync-verify`).

D.5 Office Documents (3 Skills)

Programmatic creation of PowerPoint (`pptx-create`), Word (`docx-create`), and Excel (`xlsx-create`) documents from research content.

D.6 Editorial Intelligence (4 Skills)

Diagnostic editorial tools that analyze and improve writing across contexts—papers, blogs, grants—focusing on diagnosis and translation rather than generation from scratch.

D.7 What This Demonstrates

These skills show that **discovery and verification infrastructure is buildable today**. Citation verification catches hallucinations; code-paper consistency is checkable; writing workflows preserve researcher agency while accelerating execution.

What remains missing: benchmarks for objective comparison. We cannot answer "Does our citation checker outperform alternatives?" or "Which writing skill produces clearer prose?" This gap—not the skills themselves—is our primary contribution: identifying what infrastructure the research community needs to build collectively.