Behind the Paper: AI-Assisted Research

Companion to "Can AI Ace Your Exam? A Penn Law Experiment"

R. Polk Wagner[†]

University of Pennsylvania Carey Law School

Early Draft — June 2026

The study and its apparatus

The paper this essay accompanies asks a narrow empirical question: can an artificial intelligence pass your final exam, and does handing it the course materials make it better? To answer that, I had to put a model through a controlled experiment — generate the answers, package them as ordinary Examplify exports, and send them out for blind grading alongside real students. There is an irony here that I did not engineer but cannot ignore. A study of AI sitting law exams was itself conducted with AI as the laboratory. The model was both the subject and, in a different role, the instrument.

This arrangement turned out to be perhaps the most instructive part of the project, and the part the paper itself never discusses. Running a research program with an agent as a collaborator — Claude Code, working from a project folder I controlled — forced me to be precise about something I would otherwise have left vague: what, exactly, can these tools be trusted to do, and where does the trust have to be replaced with structure?

The honest answer is that the machine's role, and its limits, looked different at three distinct stages. The work moved through running the experiment, keeping the record, and writing and verifying the paper. At each stage the agent did something real, and at each stage I had to build a 'fence' around it. This is the story of those fences — and a usable account, I hope, for any colleague who wants to try something similar.

[†]Michael A. Fitts Professor of Law at the University of Pennsylvania Carey Law School. This essay is a companion to "Can AI Ace Your Exam? A Penn Law Experiment," describing how that study was conducted with the assistance of an AI coding agent. Early working draft (June 2026); faculty and students are anonymized.

Running the experiment

Here the AI was the apparatus, not a writing aid. The design is a controlled 2 × 2: two exam questions crossed with two materials conditions. In the project's own framing, "the empirical question is whether attaching course materials (outlines, syllabus) measurably improves AI exam performance," with an A arm where "no materials attached" and the "model uses only its training knowledge," and a B arm where "course materials attached (typically a syllabus + 1-2 student outlines)." Four answer files per exam, packaged into two Examplify PDFs, one per condition.

The methodological problem with using one model to generate all four cells is obvious: a single conversation that produced Q1 with materials and then Q2 without them would leak. The no-materials arm would be tainted by everything the model had just read. So the cells never share a conversation. Each is generated by an isolated subagent, dispatched in parallel, and the rule that protects this is stated flatly in the project's invariants:

> Each of the four cells (Q1-A, Q1-B, Q2-A, Q2-B) is generated by a separate `general-purpose` subagent dispatched in parallel via the `Agent` tool with `run_in_background: true`. The orchestrator dispatches and captures only — it does not draft, edit, or summarize the substantive answer text. Cross-cell visibility breaks the design.

That last clause is the key. The orchestrating agent — the one I talk to — is deliberately walled off from the substance. It schedules the work and collects the outputs; it never decides what a good answer to the tax question is. The same logic governs what each cell is allowed to read. The materials boundary holds that "each arm sees *only* the files in its arm-specific materials folder," and that the operator-reference directory at the exam-folder root "must not be exposed to any cell," because "violating the boundary contaminates the experimental contrast." The scoping is not housekeeping. The scoping *is* the experimental contrast: the only difference between the arms is the folder each one was permitted to open.

And then the rule that runs against every instinct an agent has to be helpful: verbatim capture, then audit. Each subagent's output is saved to disk "*exactly as returned*, including any protocol violations (preambles, em dashes, etc.)." Violations get documented first and cleaned only on a "second explicit pass," because "silent first-pass cleanup destroys the research record." A model wants to tidy as it goes. A clean experiment needs the mess preserved long enough to be counted. Cleaning before recording would have quietly erased exactly the protocol-compliance data the paper would later need.

Keeping the record

The discipline that surprised me most was not in the generation at all. It was forcing the machine to maintain the research record *as the work happened*. A decision log, a lessons-learned file, a canonical CSV of every cell and score, a cross-exam dashboard — each updated on the same pass

that produced the answers, not reconstructed afterward from memory. The payoff is that the eventual Methods section writes itself from a gap-free trail rather than from my recollection of what I think I did in April.

This only works because the obligation is a rule for the agents, not an aspiration. The record "is the source from which the eventual paper's Methods section will be drafted," and keeping it current is "not optional — gaps in the record become gaps in the paper." The mechanism is a literal checklist that fires on every new exam: add a row to the summary CSV, append to the materials inventory, note any format variation, log new lessons, log new decisions. The transferable lesson is unglamorous but real: the record is only as good as the rule that says update it on every run. An agent will happily maintain a meticulous log if you make logging part of the task definition, and will just as happily skip it if you don't.

The reason this matters is best shown by a single entry. One of the logged lessons concerns whether the materials actually expand what the model can do. Across three Wave 1 exams, the answer was a clean and slightly deflating no. On one exam, both AI arms missed the same issue in the first question — an issue roughly fifteen percent of real students also missed — even though the materials arm had received the full syllabus and two student outlines covering the entire course, and the missed doctrine was standard coverage. The log records the consequence in plain terms: "Despite the materials, the B arm did not produce coverage the A arm lacked on this specific issue." The same shared-blind-spot pattern repeated on two further exams, both on the multiple-choice side, where both arms picked the same wrong answer despite holding different information.

And yet the materials were not doing nothing. They moved the *quality* of the analysis on issues both arms already recognized. The log's own conclusion: "The materials advantage Wave 1 has measured… is therefore probably driven by analysis quality on shared-recognized issues rather than coverage expansion." That sentence is the discipline paying for itself. Without a contemporaneous, item-level record I would have reported the loose, intuitive finding — materials help — and stopped there. The record forced the precise claim instead: materials improve analysis on the issues both arms see, but they do not expand the set of issues the model spots. The paper's central contribution is sharper because the log refused to let the easy version stand.

Writing and verifying

The last stage is the one most colleagues worry about, and rightly: how do you let a model help write the paper without the prose curdling into the familiar machine register? Two things made it work, and neither was trusting the model.

The first was voice. The draft was written against a profile built from a corpus of my own prior papers — twelve of them, weighted toward two I named as the primary target, *Information Wants to Be Free* and *Poisoning the Next Apple?* The profile catalogs the moves I actually make: the

"And yet" pivot, the colon that sets up a claim and then delivers it, the close that lands a one-line verdict rather than recapping the parts. Writing to a documented voice, rather than to a model's default, is the difference between a paper that sounds like me and one that sounds like everyone. (The profile is also, candidly, a check on me — it flags my own drift toward the tells, including inline bold, which it correctly identifies as not-mine.)

The second, and the load-bearing one, was verification — and verification by structure, run in parallel and adversarially, rather than by asking the model whether it had been good. Independent agents checked facts, citations, voice, and AI-tells against each other; a names registry stood between the draft and any fabricated or mis-anonymized person; a `[VERIFY]` tag discipline marked every uncertain claim for resolution before delivery; every number in the paper was traced back to the source CSV. The model was not policing itself. It was being audited by checks built around it.

Two catches show why the audit, not the author, is what you trust. The first was a factual overstatement I had let through. The draft claimed that a recent Stanford study's "*expert-validated ranking* placed Claude Opus 4.7 first among the dozen systems it compared." The verification pass read the source and found this "factually wrong as written": the human evaluation "tested only *two* systems," and a different model "ranked first"; Claude topped only "a *separate LLM-as-judge extension*." Conflating the two, the reviewer noted, "overstates the independence of the 'Claude is strongest' claim." I agreed and fixed it. That is precisely the kind of plausible, citation-dressed error that survives every casual read — and exactly what an adversarial check exists to catch.

The second was an anonymization leak the names guard caught at the level of the file, not the prose. A footnote *key* in the source — not the visible text, the markdown anchor — was a real instructor's surname, and as the reviewer put it, "anyone sent the source can `grep`" it; the key could survive into the exported Word or PDF as a live anchor. The fix was a mechanical sweep renaming every name-derived key to a neutral token. This is the whole reason the registry exists. The failure mode it guards against is the one my own global instructions describe: "fluent prose with a plausible first name reads as correct to every downstream reviewer," so a confidently wrong name "propagate[s] through multiple surfaces before anyone catches" it. A name is the one error a model states with total fluency and total confidence, and the one a reader is least equipped to question. So you do not ask the model. You check against a list.

What I'd tell a colleague

Four things, distilled from running this.

First, constrain the collaborator agents with structure, not with hope. Isolation between cells, hard materials boundaries, and a short list of invariants you will not violate silently are what protect the experiment; an instruction to "be careful" protects nothing.

Second, make the machine keep the record as it goes. A contemporaneous, rule-enforced log is not bureaucracy — it is the thing that turns a vague finding into a precise one, and it is the Methods section you will otherwise have to reconstruct from memory you do not have.

Third, verify adversarially and in parallel. Trust comes from independent checks built around the model — on facts, citations, names, and numbers traced to source — not from the model's own assurance that it got things right.

Fourth, keep the substance human. The agent dispatches, captures, logs, and audits; it does not decide what the answer is or what the paper claims. The orchestrator agent's deliberate ignorance of the substance is a feature.

The underlying study found that an AI can do creditable work on a law exam, and better work when you hand it the materials — but only on the issues it already knew to look for. Running the study with AI agent collaboration taught the companion lesson: the same scrutiny we ought to bring to an AI sitting our exams is exactly the scrutiny that made it trustworthy in the lab.

Appendix: The toolkit

For the record, and for any colleague who wants to reproduce the setup, here is the actual stack behind the project. Inference runs cloud-side, so the work was hardware-agnostic and crossed both of my machines over the spring without ceremony. There was no specialized infrastructure, no fine-tuning, and no private model.

Model and harness

- Claude Opus 4.7 (1M context) — model identifier `claude-opus-4-7[1m]`, at default sampling. Every cell of every exam used this one model; the parallel subagents inherited it from the orchestrator session with no per-cell override.
- Claude Code, Anthropic's command-line agent, run as a single interactive orchestrator session.
- The Claude Code `Agent` tool for parallel, isolated subagents (`subagent_type: general-purpose`, `run_in_background: true`) — the mechanism that kept the four cells from seeing one another.

Hardware and operating system

- A MacBook Pro (Apple M3 Pro) and a Mac Mini, both running macOS. The project files live on a shared cloud drive reachable from either machine.

Languages and document toolchain

- Python 3 (via Homebrew) for all scripting.

- Three small custom scripts: one to render each answer into the registrar's Examplify export format, one to assemble the multiple-choice answer workbook, and one to regenerate the summary table from the canonical results CSV.
- For the paper and this essay: pandoc to xelatex (TeX Live), set in Libertinus Serif, driven by one reusable build script.

Source preparation

- A document converter (PDF, Word, and PowerPoint to clean markdown) to turn course materials and hand-transcribed exam prompts into text the model could read — and, per the project's own rule, to check every hand-transcribed prompt against its source.

Verification and integrity

- A multi-agent review pipeline that fans out parallel agents to check facts, citations, and internal consistency, with an adversarial re-verification stage.
- A separate voice-and-AI-tell checker, run against a profile built from my own prior writing.
- A names registry as the single source of truth for every person named, and a `[VERIFY]`-tag discipline resolved before any document was called finished.

Record and storage

- The research record itself — a decision log, a lessons-learned file, the canonical CSVs, and a cross-exam dashboard — kept as plain markdown and CSV in the project's cloud-drive folder.
- A post-grading faculty survey administered through Qualtrics.

The novelty was never in the tools. It was in the fences I built around them.