Edit-Trace Oversight: Scalable Alignment Signal from Agentic Workflows

Vladimir Ovcharov*
LEX AI LLC, Kyiv, Ukraine
https://legal.org.ua

May 2026

Abstract

Edit-traces from production agentic workflows produce alignment signal that is denser, more outcome-predictive, and distributionally unlike conventional RLHF preference data [5, 10]. Three experiments on a single-practitioner case study (30,510 edit pairs, 2,892 sessions, 1,579 attributed outcomes) establish this empirically. **(1)** The practitioner's default mode is near-total rewrite: 80.7% of edits are substantive (median normalized edit distance 0.84), a distribution expected to differ sharply from crowd annotation. **(2)** Process-level behavioral features (keystroke timing, idle gaps, app switching) carry statistically significant signal ($p < 0.001$) but are computationally redundant with artifact-level features for prediction—simplifying instrumentation requirements. **(3)** Binary rejection of agent output is the single most informative oversight action, correlating with 78% positive downstream outcomes—far exceeding substantive rewrites (48.7%) or cosmetic edits (52.7%).

We formalize these findings as **edit-trace oversight**: alignment signal captured natively when a practitioner works agentically with an LLM under a *domain constitution*—five conditions defining when human corrections constitute valid training signal. The dataset is drawn from 105 days of production work (1,547 merged PRs, 70+ MCP tools, 380M+ records) building a legal-technology platform with Claude Code as the primary agentic counterpart. Signal is captured on two axes: artifact-level (what was corrected) and process-level (how oversight was performed), with outcome attribution at 88.1% strong confidence.

The core empirical claim: a single practitioner under product accountability generates preference signal that is impossible to obtain through detached annotation. DPO training (Experiment 4) will test whether this distributional difference translates into improved domain-specific model performance across four matched-volume conditions.

Keywords: RLHF, preference data, scalable oversight, agentic workflows, edit-trace, domain constitution, legal AI

1 Introduction: The Oversight Gap in Agentic Systems

1.1 Motivation: Empirical Observation of Recursive Human–LLM Composition

Scalable oversight research typically frames the problem defensively: as models become more capable, mechanisms must compensate for the limits of human attention—debate [6], recursive

*Correspondence: volodymyr@legal.org.ua

reward modeling [8], AI judges, constitutional methods [2, 5]. This framing treats the bandwidth of human oversight as a fixed constant and asks how to route around it.

The case study documented in this paper presents an empirical observation that complicates this framing. A single practitioner shipped 1,547 PRs across 7 production systems in 105 days using an LLM agent (Claude Code) as the primary engineering counterpart. Neither party would have reached this output independently: the practitioner's throughput without the agent is bounded by typing and cognitive load; the agent's autonomous reliability at consequential scale remains insufficient for production deployment without human oversight.

In this regime, the practitioner applies corrections at each step, and each correction shapes the context for subsequent agent actions. The resulting edit-traces are generated under two constraints that standard annotation lacks: (1) *production accountability*—the corrected output ships to real users, creating a natural incentive for concentrated attention at each decision point; and (2) *sequential dependence*—corrections accumulate along the trajectory, meaning each edit is informed by the consequences of prior corrections.

We observe empirically that this regime produces a qualitatively different distribution of corrections compared to what we would expect from detached annotation (Section 6.1). Whether this distributional difference translates into better training signal is an empirical question addressed by Experiment 4 (Section 6.4).

1.2 The Structural Problem

As LLM agents take on longer-horizon, multi-step work—composing tool calls, accumulating context across hundreds of turns, and shipping outputs with attributable real-world stakes—the gap between how we collect alignment signal and how agents actually fail has become structural. Every existing source of RLHF preference data shares one property: **the annotator operates outside the agentic workflow they are meant to govern** [1, 10, 15]. A Mechanical Turk worker rates isolated model outputs without a codebase, a deployment pipeline, or a customer on the other end. An expert annotator evaluates in a controlled environment, not mid-trajectory in a compositional system. An RLAIF model [7] applies principles supplied by its creators, without feedback from the downstream consequences of the agent's actions. They all produce ratings detached from the granularity at which agentic systems actually fail: the individual edit within a multi-step trajectory under domain constraints and outcome accountability.

1.3 Edit-Trace as Oversight Signal

We propose an alternative: **edit-trace oversight**—alignment signal captured natively when a practitioner works agentically with an LLM over consequential, multi-step workflows.

When a practitioner runs Claude Code agentically—composing tool calls, reviewing architectural proposals against domain constraints, accepting or rejecting suggested changes based on information not available to the model—every human edit on a model output is a localized correction relative to a domain constitution and an outcome trajectory. This is not preference annotation. It is **in-the-loop oversight**, captured at the granularity where agentic systems actually fail.

Two properties distinguish edit-trace oversight from expert annotation:

Outcome-validated corrections. The practitioner makes binding decisions with real consequences. Accepted agent output ships and passes or fails in production. Each edit-trace is a correction grounded in revealed preference + ground truth, not abstract judgment.

Compositional trajectory awareness. The practitioner builds compositional pipelines (Query Planner → Semantic Sectionizer → Hallucination Guard → Citation Validator), where every oversight correction affects the rest of the trajectory. Each edit encodes not just local quality judgment but awareness of how the correction propagates through downstream components. This is qualitatively more informative than isolated rating of individual model outputs.

1.4 Behavioral Context of Oversight Actions

Even rich edit-trace capture records only the artifact-level correction (*what* the practitioner changed). The cognitive and behavioral context behind the correction—time invested, external research consulted, voice calls made, window switches indicating cross-referencing—is lost. We capture this dimension through synchronized OS-level activity tracking, providing the behavioral context of each oversight action. This enables the question: does *how* a practitioner performs oversight contain signal beyond *what* they corrected?

1.5 Research Questions

RQ1 Does oversight edit-trace from an agentic practitioner differ distributionally from crowd annotation on matched LLM outputs?

RQ2 Does the behavioral context of oversight actions contain signal beyond the artifact-level correction?

RQ3 Do oversight corrections within agentic workflows correlate with downstream outcomes?

RQ4 Does training on oversight-trace preferences improve domain-specific performance vs. crowd-sourced baselines?

2 Related Work

RLHF preference collection. The dominant paradigm for aligning language models relies on human preference judgments collected in controlled settings. Christiano et al. [5] introduced pairwise comparisons over trajectory segments; Stiennon et al. [15] and Ouyang et al. [10] scaled this to natural language tasks using crowd workers and contractors. Bai et al. [1] compared the signal quality of crowd annotators versus researchers. In all cases, annotators operate outside the systems they evaluate—rating isolated outputs without access to the deployment context, downstream consequences, or the compositional trajectory that produced the output. Edit-trace oversight departs from this paradigm: the signal source is the practitioner who ships the output, not a detached evaluator.

Scalable oversight. As model capabilities grow, the cost and reliability of human oversight become central concerns. Irving et al. [6] proposed AI Safety via Debate, where models argue for and against answers to aid human judgment. Leike et al. [8] formulated recursive reward modeling, decomposing hard oversight tasks into easier subtasks. Bowman et al. [3] provided benchmarks for measuring oversight progress, and Burns et al. [4] demonstrated weak-to-strong generalization, where weaker models supervise stronger ones with partial success. These approaches treat human oversight bandwidth as a bottleneck to be routed around. The regime documented here suggests an alternative: when a practitioner works agentically with an LLM, oversight bandwidth may *scale*

with agent capability rather than against it, as the human's corrections become more targeted while the agent handles routine execution.

AI feedback, constitutional methods, and formal control structures. Bai et al. [2] introduced Constitutional AI, replacing human annotators with AI self-evaluation against researcher-authored principles. Lee et al. [7] extended this with RLAIF, showing that AI-generated feedback can approximate human preferences at lower cost. Both eliminate the annotation bottleneck but operate without production grounding—the principles and feedback are applied in abstract evaluation contexts, not during consequential deployment. A parallel tradition in knowledge engineering uses formal structures to directly control system behavior: ontology-controlled architectures [11] govern information systems through domain ontologies, and recent work applies this principle to LLMs—OntoChatGPT [12] uses formal ontologies to structure ChatGPT's output via meta-learning prompts, while Palagin et al. [13] demonstrate that integrating neural network and ontolinguistic paradigms yields stronger results than either alone. The domain constitution proposed here draws on both traditions: like Constitutional AI, it defines formal conditions for evaluating model output; like ontology-controlled architectures, it uses formal structure to govern system behavior—but applied to the oversight process rather than the generation process.

Direct preference optimization. Rafailov et al. [14] introduced DPO, which optimizes a language model directly on preference pairs without training a separate reward model. DPO's reliance on paired preferences (chosen vs. rejected completions) makes it a natural fit for edit-trace data, where each human correction provides an implicit preference pair: the practitioner's corrected output (chosen) versus the agent's original output (rejected). Experiment 4 (Section 6.4) uses DPO to test whether the distinctive distribution of edit-trace preferences translates into improved domain-specific model performance.

3 Defining Valid Oversight: The Domain Constitution

As discussed in Section 2, existing approaches to preference collection and formal AI control operate at the level of model output. We define a **domain constitution**—formal conditions under which human corrections on agentic output constitute valid oversight signal. Where Constitutional AI asks "does this output satisfy these principles?" and ontology-controlled systems govern what the system produces, the domain constitution governs *when corrections on system output constitute valid training signal*—shifting formal control from the generation process to the oversight process.

Not all human–agent interaction produces oversight signal. A user who copies an LLM snippet into a one-off script provides no oversight. A crowd annotator who rates two completions provides weak oversight, ungrounded in real consequences. The domain constitution specifies the boundary conditions that separate noise from signal.

3.1 Two-Axis Oversight Signal

Artifact-level: what was corrected between agentic output and final artifact—edit distance, semantic change class, structural changes. This axis captures the *content* of oversight: which agentic behaviors the human deemed unacceptable, and how they were remediated.

Process-level: how the correction was made—keystroke timing patterns, idle gaps, app-switching trajectory, voice context. Captured only with OS-level instrumentation running in parallel. This

axis captures the *cognitive cost* of oversight: how much effort the correction required, what external information was consulted, and whether the human deliberated or corrected reflexively.

3.2 Five Conditions of the Domain Constitution

The domain constitution specifies five conditions that must hold simultaneously for human edits on agentic output to constitute valid oversight. Each condition addresses a specific failure mode that would render the edit-trace uninformative or misleading as a training signal.

C1. Shared persistent state between human and agent. The agent operates against a continuously evolving codebase, not isolated snippets. Each session inherits state from previous sessions through the working directory, git history, file structure, and accumulated documentation. The codebase itself functions as long-term memory shared between human and agent.

Why necessary: Without persistent shared state, the human's corrections are context-free—they reflect preferences over isolated outputs rather than oversight over an evolving system. Persistent state ensures that each correction is informed by the full history of prior agent behavior and its cumulative consequences.

C2. Compositional task layering. Work decomposes into chains where one agentic session's output (committed code, architectural decision, documentation update) becomes context for subsequent sessions. The practitioner maintains persistent computational threads spanning days, weeks, or months.

Why necessary: Single-turn corrections cannot capture oversight over compositional failure modes—cases where each individual agent output appears adequate but the composition fails. When a practitioner corrects an architectural decision because it conflicts with a decision made three weeks earlier, the resulting edit-trace encodes long-range dependency information that no single-turn annotation scheme can capture.

C3. Grounding in observable reality. The practitioner establishes definition-of-success parameters before initiating work on any non-trivial task: what observable behavior constitutes completion, what failure modes invalidate the approach, what performance characteristics the artifact must exhibit in production.

Why necessary: Oversight that rests on subjective preference alone is indistinguishable from taste. When corrections are grounded in observable system behavior—a deployment that failed, a latency spike, an error rate increase—the edit-trace encodes causal information about what works and what does not.

C4. Information asymmetry favoring the human. The practitioner reviews every commit, evaluates architectural proposals against domain constraints, accepts or rejects suggested changes based on information not available to the agent (business priorities, regulatory requirements, user feedback, personal stake in outcomes).

Why necessary: Oversight is meaningful precisely because the overseer holds information the overseen system lacks. If the human's corrections reflect only information already available to the agent, the edit-trace is redundant with the agent's own uncertainty.

C5. Consequential grounding. The output is shippable code that runs in production with measurable consequences: feature usage, system reliability, customer adoption, revenue, partnership formation.

Why necessary: Oversight signal must connect to real consequences to avoid the same detachment that afflicts crowd annotation. When corrected artifacts ship and succeed or fail in production, the edit-trace acquires outcome labels that close the loop between correction and consequence.

3.3 Instantiation by the Case Study

This domain constitution is instantiated by the author's production work: **1,547 merged PRs across 7 interconnected projects over 105 days** using Claude Code as primary agentic counterpart. The core platform (Legal.org.ua, 1,393 PRs) produces a deployed legal AI platform with 380M+ records pipeline and 70+ MCP tools. Satellite projects (154 PRs) cover due diligence intelligence (SneakyPiper, 73 PRs), LinkedIn lead automation (aipromo, 39 PRs), meeting scheduling (Calendary, 27 PRs), OSINT aggregation (Panoptic, 10 PRs), and OS-level activity tracking (XSISTANT, 5 PRs). Measurable downstream outcomes include selection by Google for Startups, introduction to Deloitte via GFS, and acceptance into NVIDIA Inception Program.

Each of these acceptances was achieved through written applications without prior voice conversations, in-person meetings, or warm introductions from accelerator mentor networks. The applications themselves were drafted using the same recursive workflow that produced the underlying product, demonstrating that the workflow generalizes from code production to high-stakes written communication with measurable institutional gatekeepers.

3.4 What Fails to Constitute Valid Oversight

The domain constitution also defines its negation—interaction patterns that fail one or more conditions and therefore do not produce valid oversight signal:

- One-shot code generation (fails C1–C2: no persistent state, no compositional layering)
- Automated CI/CD pipelines using Claude Code (fails C4: no human information asymmetry)
- Tutorial or learning use (fails C5: no consequential grounding)
- Pair programming without pre-defined success criteria or observable production feedback (fails C3)

3.5 Edit Taxonomy

Six semantic change classes: `cosmetic`, `reorganization`, `factual_correction`, `tone_adjustment`, `substantive_rewrite`, `rejection`. Classification is two-phase: rule-based boundaries (edit_distance_norm < 0.05 = cosmetic, ≥ 0.80 = substantive_rewrite), then Claude Sonnet 4.6 via AWS Bedrock for the ambiguous middle range. Coverage: 99.96%.

4 Data Collection Architecture

4.1 Workflow-Level Capture

Three retrospective extractors feed the `rlhf-signals` module:

- **GitHub PRs** (GraphQL API)—commits, diffs, review comments, merge status.
- **Plane issues** (REST API)—state transitions, comment threads, domain problem refinement.
- **Claude Code transcripts** (local JSONL)—richest source, avg. 26.8 artifacts/session.

Schema: `workflow_sessions` → `workflow_artifacts` → `workflow_edits` → `workflow_outcomes`.

GitHub PR velocity (core platform): 1,393 merged PRs over 105 days (87 active). Peak: March 790 PRs (25.5/day). Median time-to-merge: 30 seconds (77.8% under 5 min)—solo-practitioner auto-merge pattern. PR timestamps do *not* reflect editing time; real duration is reconstructed from OS-level activity.

4.1.1 One Shipping Operation, Multiple Technical Surfaces

These are not separate projects in different business domains. They are **components of one shipping operation**—making Legal.org.ua succeed—spanning different **technical surfaces**. All 1,547 PRs serve one outcome: the platform works, has paying customers, and wins institutional validation.

Technical surfaces within the core platform (1,393 PRs): frontend (React 19, Vite, TailwindCSS), backend (Express, MCP protocol, 70+ tool handlers), data engineering (court decision harvesting, 380M+ records), database (PostgreSQL migrations, Qdrant vector indexing, Redis caching), DevOps (Docker, nginx, CI/CD, blue-green deployment), content (blog, SSG, SEO), and shared TypeScript packages.

Table 1: Satellite components within the unified shipping operation.

Component	**PRs**	**Period**	**Role in Legal.org.ua**
SneakyPiper	73	Apr 14–30 (17d)	Due diligence on SecondLayer API
aipromo	39	Apr 5–May 3 (29d)	LinkedIn leads via Bedrock + Unipile
Calendary	27	Apr 13–14 (2d)	Meeting scheduling for leads
Panoptic	10	Apr 7–16 (10d)	Threat intel feeding SneakyPiper
XSISTANT	5	Apr 18 (1d)	Activity tracker for this research
Total	**1,547**	**105 days**	**7 repos, 1 practitioner, 0 employees**

4.2 OS-Level Activity Instrumentation

Parallel to workflow tracking, an OS-level activity tracker records 5-second activity buckets:

- `activity_scores`—active/passive/idle classification + keystroke/mouse/click counts.
- `input_activity`—keystroke and mouse counts per 5s bucket (never keystroke *content*).
- `window_sessions`—focused app + window title + working directory.
- `idle_events`—gaps without input, with duration.
- `mic_activity`—voice/call context detection.

Storage: ∼38 MB for 21 continuous days. Both databases store `timestamptz` in UTC—cross-source alignment verified to <3 seconds.

4.3 Cross-Source Linking

For each edit with time window $[T_1, T_2]$: query activity in $[T_1 - 30\text{s}, T_2 + 30\text{s}]$, aggregate process features, classify window sessions by category (`code_editing` / `research` / `communication` / `documentation` / `unrelated`).

4.4 Practitioner Disambiguation Sessions

Automated activity tracking captures *what* app or window was active, but cannot determine *why*. A YouTube video about Ukrainian court procedure, a YouTube video about astrophysics, and a YouTube video about cooking all look identical in the activity data—same `wm_class`, similar engagement metrics. Yet their relationship to the subsequent editing session is fundamentally different.

We address this through **periodic practitioner disambiguation sessions**—structured interviews where the practitioner reviews ambiguous activity windows from the preceding period and provides ground-truth labels. Ambiguous activity categories requiring practitioner input:

(a) **On-topic research.** Task-relevant content consumption (*e.g.* a conference talk about vector databases before refactoring the Qdrant pipeline).
(b) **Cross-topic inspiration.** Cross-domain intellectual intake that influenced subsequent editing quality.
(c) **Conversational reasoning with Claude.** Deliberative reasoning via claude.ai or Claude Code in conversational mode—not producing commits, but shaping architectural decisions.
(d) **Genuinely unrelated activity.** Social media, personal messaging, entertainment.

These labels feed back into the `workflow_edit_engagement` table as a `disambiguation_label` field, enabling more accurate window category computation and a novel process feature: `cross_topic_inspiration_`

4.5 Outcome Attribution

Automated attribution with confidence levels: **strong** (temporally proximate, causally linkable—PR merged, no revert in 30d), **medium** (present but confounded), **weak** (causally tenuous).

5 Verified Pilot Dataset

All numbers verified from production databases as of May 8, 2026.

5.1 Workflow Data

Table 2: Pilot dataset summary (`rlhf-signals` database).

Metric	**Value**
Workflow sessions	2,892 (Jan 26–May 8, 2026)
Sources	Claude Code: 1,051 • GitHub PRs: 1,013 • Plane: 828
Total artifacts	33,402 (21,477 LLM outputs)
Total edit pairs	30,510
Attributed outcomes	1,579 (54.6% session coverage)
Outcome confidence	88.1% strong, 5.1% medium, 6.8% weak

5.2 Edit Distribution (Oversight Signal)

Edit distance (normalized): mean = 0.807, median = **0.839**, P25 = 0.743, P75 = 0.927, P95 = 0.987. The practitioner's default mode is near-total rewrite.

Table 3: Distribution of edit-trace oversight across semantic change classes.

Semantic Class	Count	%
Substantive rewrite	24,619	80.7
Reorganization	2,106	6.9
Cosmetic	2,098	6.9
Rejection	1,110	3.6
Factual correction	307	1.0
Tone adjustment	259	0.8

5.3 Process-Level Data

Table 4: OS-level activity instrumentation data (XSISTANT database).

Table	Rows	Coverage
`activity_scores`	52,451	Apr 17–May 8 UTC
`input_activity`	46,543	176K keystrokes, 52% in terminal
`window_sessions`	16,901	App + title + working directory
`idle_events`	54,974	Avg idle: 51 min
`mic_activity`	1,869	Apr 16–May 3

Bimodal work pattern: 07–11 UTC primary peak (1,376 active windows), 19–21 UTC secondary peak. Approximately 13% real engagement time (6.7% active, 6.3% passive, 87% idle).

5.4 Overlap Window

Table 5: Overlap between XSISTANT instrumentation and `rlhf-signals` data.

Metric	Value
Sessions in overlap	498 (17.2%)
Edits in overlap	9,254 (30.3%)
GitHub PRs in overlap	75 / 1,393 (5.4%)
Outcomes in overlap	64

The main PR burst (Feb–Mar, 1,156 PRs at 25.5/day) occurred **before** XSISTANT launched. Process-level enrichment covers steady-state work (4.4 PRs/day), not peak sprint.

6 Experiments

Four experiments with progressively higher compute requirements. Experiments 1–3 require no GPU.

6.1 Experiment 1: Oversight vs. Annotation—Distributional Difference (RQ1)

Status: Phase A complete.

Sample $N = 200$ LLM outputs (stratified by semantic class, min 10 per class), send to crowd annotation platform. Compare in-the-loop oversight corrections vs. detached crowd annotation: edit distance distributions (KS test), semantic class breakdown, inter-annotator agreement (Krippendorff's α). The central question is whether corrections applied during live agentic workflows differ *in kind* from labels applied after the fact.

Phase A results (sampling completed May 8, 2026): 19,455 eligible samples after PII filtering (from 21,461). Stratified allocation: substantive=144, cosmetic=15, reorganization=11, rejection=10, factual=10, tone=10. Two JSONL exports: full metadata + platform-ready (no oversight edits shown to annotators). Deterministic seeded PRNG for reproducibility.

Table 6: Experiment 1 Phase A stratified sampling allocation.

Class	**Sampled**	**Population**
substantive_rewrite	144	15,762
cosmetic	15	1,676
reorganization	11	1,148
rejection	10	440
factual_correction	10	217
tone_adjustment	10	212

Expected: Oversight corrections show heavier tail (80.7% substantive_rewrite already—crowd annotators, operating without production context, are unlikely to match this intensity).

6.2 Experiment 2: Behavioral Context of Oversight Actions (RQ2)

Two predictive models on the 498-session overlap subset: Model A (artifact-only) vs. Model B (artifact + behavioral-context features). Compare AUC, SHAP feature importance, permutation test. With only 64 outcomes in overlap, we use edit-class proxy labels for statistical power.

Cross-source linking. Joined XSISTANT OS-level activity data (52,272 activity scores, 16,122 window sessions) with workflow edits via artifact timestamps. Alignment verified to <3s. Result: 10,846 edits processed; 6,753 (62.3%) with process data.

Process features computed per edit: active/passive/idle seconds, keystroke counts, mouse distance, idle gap analysis, app switching count, research switches, voice context, window dwell entropy, window category seconds.

Model comparison. Target variable: binary—substantive_rewrite (1) vs. cosmetic (0). $N =$ 6,152 edits (5,740 substantive, 412 cosmetic). 5-fold stratified cross-validation.

Table 7: Experiment 2 model comparison: artifact-only vs. artifact + behavioral-context.

Model	**Random Forest AUC**	**Logistic Regression AUC**
A (artifact-only)	$\mathbf{0.903 \pm 0.005}$	0.475 ± 0.018
B (artifact + behavioral)	0.874 ± 0.007	$\mathbf{0.540 \pm 0.039}$
Delta (B − A)	−0.029	+0.065

Permutation test (1,000 iterations, behavioral-context features shuffled): $p < 0.001$—behavioral-context features carry statistically significant, non-random signal.

Paired t-test (RF, 5 folds): $p = 0.003$—the delta is statistically significant (in the negative direction for RF).

Interpretation. Behavioral-context signal is **real and non-random** (permutation $p < 0.001$), but does not improve Random Forest prediction of edit class. This is a nuanced result: (1) The proxy target is already well-predicted by artifact features alone (AUC 0.903), leaving little room for improvement. The 14:1 class imbalance further limits discriminative contribution. (2) Behavioral-context features help linear models (+0.065 AUC), suggesting the signal exists but is captured non-linearly by artifact features in RF. (3) The real test requires real outcomes—only 64 outcomes exist in the overlap window, insufficient for outcome prediction.

Behavioral-context signal exists (permutation proof) but is largely redundant with artifact signal at this scale and target definition. This suggests that artifact-level capture of oversight corrections is sufficient for most preference learning, and behavioral context adds value primarily for edge cases or different prediction targets.

6.3 Experiment 3: Oversight Corrections and Downstream Outcomes (RQ3)

6.3.1 Level 1: Full Dataset (Artifact-Only)

30,499 edits joined with 1,579 outcomes.

Table 8: Experiment 3 correlations (full dataset).

Metric	**Value**
edit_distance_norm ↔ outcome	$r = -0.116$, $p < 0.001$
semantic_class ↔ outcome	$r = -0.137$, $p < 0.001$
KS test (positive vs. negative)	0.253, $p < 0.001$

Table 9: Positive outcome rates by edit class.

Class	**Positive Rate**	N
Rejection	**78.0%**	431
Cosmetic	52.7%	383
Substantive rewrite	48.7%	3,692

Key finding: Rejection (completely halting the agentic trajectory) has the highest positive outcome rate. This suggests the most valuable oversight signal is the binary accept/halt decision, not granular edit depth. The overseer's willingness to say "no, start over" is more predictive of good outcomes than careful correction. This has direct implications for scalable oversight: **the single highest-value data point is whether a human stopped the agent.**

Negative correlation of edit distance with outcomes: smaller corrections correlate with better outcomes ($r = -0.116$). When the agent's output is already close to what the overseer needs, the final product tends to succeed. Heavy rewrites may indicate the agent was on the wrong trajectory.

6.3.2 Level 2: Overlap Subset (Artifact + Behavioral Context)

807 edits with both behavioral-context data and outcomes (720 with binary positive/negative label). Engagement quartile analysis shows Q4 (highest engagement) has visible positive outcome enrichment compared to Q1–Q3, but the effect is modest. The small sample size limits statistical power for definitive engagement–outcome claims.

6.3.3 Confound Controls

Hour of day, day of week, and session source all affect outcome rates independently of edit patterns. The bimodal work schedule (07–11 UTC peak, 19–21 UTC secondary) introduces temporal confounds that must be controlled in any outcome prediction model.

6.4 Experiment 4: Training on Oversight-Trace vs. Annotation Preferences (RQ4)

Redesigned based on Experiments 1–3 findings (see Section 7).

The flagship experiment requiring GPU compute. Simplified from 5 to 4 core conditions after Experiments 2–3 showed behavioral-context weighting is unlikely to improve over uniform weighting.

Four training conditions on Llama 3.1 8B or Qwen 2.5 7B (open-weight):

Table 10: Experiment 4 training conditions (revised design).

Cond.	Description	Rationale
A	Oversight-trace, uniform (24,495 pairs)	Practitioner edits—the distinctive distr of corrections from production agentic wo
C	RLAIF self-correction (24,495 matched pairs)	AI self-correction baseline—tests whether oversight captures signal that AI self-eva misses
E	Public RLHF baseline (UltraFeedback, 24,495 sampled pairs)	General-purpose RLHF baseline—tests specific value of edit-trace signal
D	Untrained instruct model	No-preference baseline

Method: DPO [14]. Evaluation: win-rate (GPT-4 judge + human $N = 100$), domain accuracy, AlpacaEval 2.0, length-controlled win rate.

Primary metrics—three comparisons: A vs. D (edit-trace improves stock model), A vs. C (human oversight vs. AI self-correction), A vs. E (domain-specific vs. general RLHF).

Estimated cost: $310–380 (see Appendix A).

7 Cross-Experiment Synthesis

7.1 The Three Findings

Finding 1 (Exp 1): Oversight corrections are qualitatively different from annotation. 80.7% of all edits are substantive rewrites. Median normalized edit distance is 0.84—the overseer's default mode is near-total rewrite of LLM output. This distribution will almost certainly differ from crowd annotators, who—operating without production context or domain stakes—tend toward safe, cosmetic edits.

Finding 2 (Exp 2): Behavioral context is methodologically important but computationally redundant. Permutation testing confirms behavioral-context features carry statistically significant signal ($p < 0.001$). However, they do not improve Random Forest prediction of edit class beyond what token counts alone achieve (AUC 0.903 $\rightarrow$ 0.874, actually worse). Behavioral-context features help linear models (+0.065 AUC) but are captured non-linearly by artifact features in tree-based models. The behavioral-context axis is a contribution to methodology, not to prediction performance.

Finding 3 (Exp 3): The most valuable oversight action is halt/reject. Completely rejecting LLM output—halting the agentic trajectory—correlates with 78% positive outcomes, far higher than substantive rewrites (48.7%) or cosmetic edits (52.7%). Edit distance negatively correlates with outcomes ($r = -0.116$): the less the overseer changes, the better the result. **The most informative oversight signal is binary (accept vs. halt), not continuous (edit distance).**

7.2 Implications for DPO Training

The original Experiment 4 design had 5 training conditions, with the primary hypothesis being that behavioral-context-weighted preferences (Condition B) would outperform uniform-weighted (Condition A). The data now challenges this:

Behavioral-context weighting is unlikely to help. Experiment 2 showed behavioral-context features do not improve prediction. Experiment 3 showed engagement quartiles barely differentiate outcomes. The α-weighted DPO formula (weight $= 1 + \alpha \cdot$ engagement_score) would scale pairs by a signal that is statistically real but practically redundant with artifact features.

The real value is in the distribution, not the weighting. The overseer's 80.7% substantive rewrite rate and 3.6% rejection rate create a fundamentally different preference distribution than either AI self-correction or general-purpose RLHF data. Training on this distribution (even with uniform weights) should produce different model behavior than training on RLAIF or public preference data. Replacing crowd annotation with RLAIF (Condition C) and public RLHF data (Condition E) enables matched-volume comparison (24,495 pairs per condition) and eliminates the annotation bottleneck.

7.3 Summary

The synthesis of Experiments 1–3 yields a clear primary contribution: oversight-trace—the edit signal captured during production agentic workflows under a domain constitution—constitutes a fundamentally different preference distribution than detached annotation. The distributional difference (80.7% substantive rewrites, 78% halt-positive rate) is substantial. Whether this distributional difference translates into improved domain-specific model performance is the subject of Experiment 4.

The behavioral-context null result is itself a contribution: it shows that capturing *what* was corrected is more informative than *how* the correction was performed, simplifying the instrumentation requirements for future deployments of this methodology.

8 Threats to Validity

The central methodological challenge of this work is that the study subject, the sole annotator, and the author are the same person. We explicitly enumerate the resulting threats and the mitigations

available within a single-practitioner protocol.

8.1 Construct Validity: Oversight Signal vs. Practitioner Skill

The domain constitution (Section 3.2) claims to define conditions under which edit-traces constitute valid *oversight* signal. However, the observed edit distribution (80.7% substantive rewrites, median edit distance 0.84) could alternatively reflect practitioner-specific editing style rather than a general property of in-the-loop oversight. That is: a different practitioner meeting all five constitutional conditions might produce a fundamentally different edit distribution.

Mitigation: The multi-practitioner cohort (Section 10) is designed specifically to disentangle practitioner-specific style from constitution-induced oversight properties. Within the current single-subject study, we note that the edit distribution is consistent across 7 technically diverse repositories and multiple technical surfaces (frontend, backend, data engineering, DevOps, content), suggesting the pattern reflects the workflow regime rather than a single domain skill. However, this remains a conjecture until replicated with additional subjects.

8.2 Internal Validity: Outcome Metrics

The outcomes cited (Google for Startups acceptance, NVIDIA Inception, paying customers) validate the *product*, not the *methodology*. They demonstrate that the recursive workflow produces shippable software, but do not directly establish that the extracted edit-traces are superior training signal compared to crowd annotation. This distinction is critical: positive product outcomes are a necessary condition for the edit-traces to carry meaningful signal (per Condition C5), but are not sufficient evidence that training on those traces improves downstream model performance.

Mitigation: Experiment 4 (Section 6.4) is designed to test the methodological claim directly: does DPO training on edit-trace preferences (Condition A) outperform training on crowd-sourced preferences of matched volume (Condition C) on domain-specific evaluation? Until Experiment 4 completes, the product outcomes serve only as evidence that Condition C5 (consequential grounding) is satisfied, not as evidence of the edit-trace's superiority as training signal.

8.3 Selection Bias: Survivorship in the Dataset

The dataset contains only edit-traces from trajectories that culminated in merged PRs and working features. Agent outputs that the practitioner accepted without correction but that later caused production failures are absent—there is no record of oversight that *should have* occurred but did not. Similarly, abandoned trajectories (work started but not completed) are systematically excluded by the retrospective extraction pipeline, which keys on merged PRs and resolved issues.

Mitigation: The survivorship bias is partially addressed by Experiment 3's finding that rejection (halting the agent) correlates with 78% positive outcomes. This demonstrates that the dataset does capture instances where the practitioner stopped an unproductive trajectory—a form of negative signal. However, the absence of false-negative oversight (accepted outputs that later failed) remains a structural limitation. Future work on failed oversight trajectories (Section 10) is designed to address this gap directly.

8.4 External Validity: Generalizability Beyond $N = 1$

This study establishes a protocol and demonstrates its feasibility with one practitioner in one domain (legal AI). No population-level claims are made or implied. The domain constitution is domain-specific by design (Section 3.2), and we expect different practitioners to require different

constitutional instantiations. Cross-domain generalizability is an explicit non-goal of the current work; the contribution is the methodology for capturing and validating edit-trace oversight, not its universal applicability.

8.5 Confound: Temporal Autocorrelation

Edit-traces within the same session are temporally autocorrelated: corrections early in a session shape the context for later corrections. Treating individual edit pairs as independent samples (as in Experiments 1–3) may overstate statistical significance. We report this as a known confound; future work should explore session-level modeling or hierarchical approaches that respect the nested structure (edits within sessions within projects).

9 Limitations

(i) **Single-practitioner protocol demonstration.** One practitioner's oversight trace, not population-level claims. Proving that edit-trace constitutes valid oversight requires a deep instrumentation case study before cohort scaling; this work establishes the methodology and qualifying criteria, not their generalizability across overseers. A multi-practitioner cohort is explicit future work.

(ii) **Domain-specific constitution by design.** The five qualifying conditions form a domain constitution that is inherently domain-specific. Legal AI oversight patterns (cross-referencing legislation, verifying citation chains) may not transfer to coding, creative, or scientific domains. This is a structural feature of constitution-based oversight, not an incidental limitation.

(iii) **Oversight captured only from successful trajectories.** The dataset contains corrections applied by an overseer whose product shipped successfully. Failed oversight—agent outputs that passed without correction and later caused failures—is systematically absent.

(iv) **Behavioral context coverage:** only 17.2% of sessions / 30.3% of edits have process-level behavioral enrichment (XSISTANT launched after peak sprint). The highest-velocity period (Feb–Mar, 1,156 PRs at 25.5/day) has artifact-level data only.

(v) **edit_seconds = 0 for all retro-extracted edits.** Timing reconstructed from 5-second OS activity buckets. Oversight action duration is bounded by bucket granularity, which may obscure rapid correction sequences.

(vi) **Structural conflict of interest:** the study subject is the overseer. In scalable oversight methodology this is structural—the overseer's corrections are the data, and the overseer's identity cannot be separated from the oversight signal. Mitigated by: external baselines, open data release, and multi-practitioner cohort design.

(vii) **Keystroke content boundary:** we capture keystroke counts and timing, never content. This trades fine-grained edit-trace information for PII protection and multi-practitioner scalability—a deliberate design constraint.

10 Future Work

- **Multi-practitioner cohort with diverse domain constitutions** (5–10 shipping practitioners across legal AI, healthcare AI, fintech, dev tools, creative tools)—each practitioner brings

their own domain constitution, enabling cross-constitution comparison and meta-constitutional analysis.

- **Failed oversight trajectories:** studying cases where oversight was absent or insufficient—practitioners who shipped without correcting agent outputs that later caused production failures. This complements the current dataset's successful-oversight bias.
- **Richer behavioral context:** eye-tracking integration captures reading-while-evaluating, the cognitive phase preceding oversight corrections, still without content capture.
- **Oversight-aware reward modeling:** an explicit reward model trained to distinguish oversight corrections from routine annotation.
- **Edit-trace capture for non-code workflows:** extending instrumentation to document drafting, legal brief composition, research analysis, and multi-turn conversational oversight.
- **Constitutional alignment between domain and model constitutions:** studying how domain-specific oversight constitutions interact with the model's own constitutional training [2] and with ontology-driven dialogue architectures [9] that use formal knowledge structures to govern human–AI interaction at the system level.
- **Automated oversight difficulty estimation:** using the edit-trace to identify which agentic steps are hardest to oversee (highest edit distance, most rejections, longest deliberation time). This produces a per-step oversight difficulty map that informs where human oversight should be concentrated.

11 Discussion

11.1 Composition as an Alternative Frame for Scalable Oversight

The empirical pattern documented here—one practitioner achieving output that neither human nor agent would reach independently—suggests a framing for scalable oversight that differs from the standard capability-gap model. In the standard model, oversight is a problem that grows harder as agents grow more capable [3, 4]. In the regime observed here, the agent's capability is not a threat to be managed but a throughput multiplier whose output the practitioner corrects at each step.

We note—speculatively—that this may represent a distinct equilibrium: rather than moving toward full autonomy as agents improve, the practitioner-agent composition may deepen, with the human's corrections becoming more targeted and architecturally informed as the agent handles more routine execution. Whether this equilibrium is stable under further capability scaling is an open empirical question that this single-subject study cannot resolve.

11.2 The Edit-Trace as Minimal Viable Oversight Signal

Experiment 3's finding—that binary rejection (78% positive outcome rate) is more informative than continuous edit distance—has practical implications. If the highest-value oversight signal is whether a human stopped the agent, then scalable oversight instrumentation may be simpler than expected: a binary accept/reject log per trajectory step, with outcome tracking, may capture the majority of useful preference signal. This hypothesis is testable in Experiment 4 by comparing DPO training on full edit-traces vs. binary accept/reject pairs.

11.3 Generalizability Beyond Software Engineering

The primary dataset was collected during software engineering workflows (TypeScript monorepo, infrastructure operations, CI/CD pipelines). A natural concern is whether edit-trace oversight generalizes to other practitioner roles.

Concurrent with the DPO training experiment (Section 6.4), the same practitioner used the identical Claude Code environment for qualitatively different tasks: ML experiment orchestration (SageMaker job configuration, DeepSpeed/LoRA hyperparameter debugging, training monitoring), academic writing (LaTeX paper drafting, BibTeX management, revision cycles), and research operations (arXiv submission workflow, academic outreach, data analysis). Over a 31-day window (April 12–May 13, 2026), automated usage instrumentation recorded 284 sessions, 492 commits, and 1,853 hours of human–agent interaction across these mixed roles.

The edit patterns observed in ML engineering sessions—rejecting incorrect DeepSpeed configurations, rewriting training hyperparameters, correcting CUDA memory management strategies—are structurally identical to the software engineering corrections in the primary dataset: the practitioner observes agent output, judges correctness against domain knowledge, and intervenes when the output fails to meet implicit quality criteria. Similarly, academic writing sessions produce edit-traces where the practitioner rewrites LLM-drafted paragraphs, restructures arguments, and corrects domain-specific claims—generating preference pairs that reflect expertise-grounded oversight rather than stylistic preference.

This observation does not constitute formal validation (a multi-practitioner, multi-domain study is needed; see Section 10), but it provides preliminary evidence that the edit-trace methodology is *role-agnostic*: it captures oversight signal wherever a domain expert iteratively corrects agent output, regardless of whether the domain is code, infrastructure, or prose.

11.4 Scope of Claims

This paper makes a methodological contribution (how to capture and validate edit-trace oversight) and reports empirical findings from a single case study (Experiments 1–3). It does *not* claim that edit-trace oversight is universally superior to RLAIF self-correction or general-purpose RLHF data—that claim requires Experiment 4's completion (where Condition C tests against AI self-correction rather than crowd annotation) and replication across multiple practitioners. The observed distributional difference between edit-trace corrections and expected crowd annotation patterns is a descriptive finding; its downstream utility for RLHF training remains to be demonstrated.

12 Conclusion

We present edit-trace oversight as a methodology for capturing alignment signal natively from production agentic workflows. The key empirical findings from a single-practitioner case study are: (1) the edit distribution under production accountability is extreme (80.7% substantive rewrites, median edit distance 0.84), and is expected to differ significantly from crowd annotation (Experiment 1, Phase B pending); (2) behavioral-context features (keystroke timing, idle gaps, app switching) carry statistically significant signal ($p < 0.001$) but are largely redundant with artifact-level features for predictive tasks (Experiment 2); (3) binary rejection of agent output is the single most informative oversight action, correlating with 78% positive downstream outcomes (Experiment 3).

The proposed domain constitution—five formal conditions under which edit-traces constitute valid oversight—provides a framework for extending this methodology to multi-practitioner cohorts.

Whether the observed distributional difference between edit-trace oversight and crowd annotation translates into improved model performance via DPO training (Experiment 4) remains an open empirical question.

The dataset (30,510 edit pairs, 2,892 sessions, 1,579 attributed outcomes) and instrumentation code will be released publicly upon completion of Experiment 4 and PII review.

A Compute Budget and Project Status

Experiment 4 requires GPU compute for DPO training. Estimated budget:

Table 11: Estimated compute budget for remaining experiments.

Component	**Cost**
DPO training (4 conditions × 7–8B model)	\$140–200
RLAIF generation (Bedrock)	\$70–80
Evaluation (judge runs, MT-Bench)	\$100
Total	**\$310–380**

As of May 2026: Experiments 1–3 complete. Experiment 4 (DPO training with 4 conditions) is pending compute allocation.

References

[1] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022. URL https://arxiv.org/abs/2204.05862.

[2] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022. URL https://arxiv.org/abs/2212.08073.

[3] Samuel R. Bowman, Jeeyoon Hyun, Ethan Perez, Edwin Chen, Craig Pettit, Scott Heiner, Kaja Lukšić, Teresa Nguyen, Amanda Askell, Kamal Ndousse, et al. Measuring progress on scalable oversight for large language models. *arXiv preprint arXiv:2211.03540*, 2022. URL https://arxiv.org/abs/2211.03540.

[4] Collin Burns, Haotian Haber, Ajeya Khoja, Emily Chen, Tadayo Treesatayapun, Saurabh Arora, and Samuel R. Bowman. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*, 2023. URL https://arxiv.org/abs/2312.09390.

[5] Paul F. Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://arxiv.org/abs/1706.03741.

[6] Geoffrey Irving, Paul Christiano, and Dario Amodei. AI safety via debate. *arXiv preprint arXiv:1805.00899*, 2018. URL https://arxiv.org/abs/1805.00899.

[7] Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. RLAIF: Scaling reinforcement learning from human feedback with AI feedback. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.00267.

[8] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: A research direction. *arXiv preprint arXiv:1811.07871*, 2018. URL https://arxiv.org/abs/1811.07871.

[9] Anna Litvin, Oleksandr Palagin, Vladislav Kaverinskiy, and Kyrylo Malakhov. Ontology-driven development of dialogue systems. *South African Computer Journal*, 35(1), 2023. doi: 10.18489/sacj.v35i1.1233.

[10] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2203.02155.

[11] Alexander V. Palagin. Architecture of ontology-controlled computer systems. *Cybernetics and Systems Analysis*, 42(2):254–264, 2006. doi: 10.1007/s10559-006-0061-z.

[12] Oleksandr Palagin, Vladislav Kaverinskiy, Anna Litvin, and Kyrylo Malakhov. OntoChatGPT information system: Ontology-driven structured prompts for ChatGPT meta-learning. *International Journal of Computing*, 22(2):170–183, 2023. URL https://arxiv.org/abs/2307.05082.

[13] Oleksandr Palagin, Vladislav Kaverinskiy, and Kyrylo Malakhov. Fundamentals of the integrated use of neural network and ontolinguistic paradigms: A comprehensive approach. *Cybernetics and Systems Analysis*, 60:111–123, 2024. doi: 10.1007/s10559-024-00652-z.

[14] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2305.18290.

[15] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://arxiv.org/abs/2009.01325.