AgentPulse: A Continuous Multi-Signal Framework for Evaluating AI Agents in Deployment

Anonymous Author(s)
Affiliation
Address
email

Abstract

Static benchmarks measure what AI agents can do at a fixed point in time but not how they are adopted, maintained, or experienced in deployment. We introduce **AgentPulse**, a continuous evaluation framework scoring 50 agents across 10 workload categories along four factors (Benchmark Performance, Adoption Signals, Community Sentiment, and Ecosystem Health) aggregated from 18 real-time signals across GitHub, package registries, IDE marketplaces, social platforms, and benchmark leaderboards. Our headline finding, on n=34 public-repo agents under a circularity-controlled test using only the Benchmark factor, is that benchmark capability predicts whether an agent ships through public IDE marketplace extensions, replicated across three independent platforms (VS Code Marketplace, Open VSX, JetBrains Marketplace): Mann–Whitney U comparing B-scores of marketplace-present vs. marketplace-absent agents gives p=0.011, 0.0011, 0.004 with rank-biserial effect sizes $+0.60$, $+0.86$, $+0.71$. The split is between agents that publish public extensions and those that do not (a heterogeneous group spanning closed-distribution agents, hosted services, and frameworks); the finding is therefore narrower than "commercialization" broadly construed. VS Code was pre-specified in the v6 draft; Open VSX and JetBrains are post-hoc external replications. The same predictor is directionally negative on library-reuse (GitHub Dependents B-only ρ_s=-0.30, B+S ρ_s=-0.38, p=0.04), evidence of discriminative validity: the framework distinguishes "agent that completes tasks" from "library that gets reused." AgentPulse is a methodology for surfacing deployment signal absent from benchmarks, not a ground-truth ranking. The framework, signals, scoring outputs, and evaluation harness are released under CC BY 4.0.

1 Introduction

AI agents, systems that combine language model reasoning (17; 19; 20) with tool use (38) and multi-step planning (30), have moved from research prototypes to production tools used by millions of developers. Existing evaluation, however, remains anchored in static benchmarks: SWE-bench (6), GAIA (10), WebArena (16), AgentBench (8), and HumanEval+ (3) all measure capability at fixed points in time. These benchmarks are essential but incomplete: they cannot capture how agents evolve through frequent updates, how reliability varies under real-world load, or how developer experience changes as workflows adapt to these tools (91).

Consistent with the ED Track's framing of evaluation as a scientific object of study, this paper investigates evaluation methodology for deployed AI agents. We ask: does a deployment-aware composite of public signals capture information benchmarks miss, and can such a composite be validated independently of the signals it aggregates? Our contribution is a measurement framework with three validation analyses plus a diagnostic, not a leaderboard.

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

We motivate this question through two concrete limitations of current agent evaluation:

Limitation 1: Existing benchmarks measure agents at fixed points along narrow capability slices, missing both the weekly release cadence of agentic systems and cross-tool comparability between, e.g., a coding copilot and a multi-agent framework whose workflows differ.

Limitation 2: Capability $\neq$ adoption. An agent's task-completion score does not predict which tools developers actually choose. Adoption depends on integration quality, pricing, reliability, documentation, and community, dimensions invisible to benchmarks.

Empirical findings preview. Three validation analyses plus one diagnostic ground the framework: (i) the four factors capture largely complementary information (n=50; $\rho_{\max}$=0.75 for Adoption–Ecosystem, all other pairwise $|\rho| \leq 0.34$); (ii) within the n=34 public-repo stratum, Benchmark capability predicts the public-IDE-marketplace shipping decision across three independent platforms (Mann–Whitney U on B-scores for present vs. absent agents: p=0.011, 0.0011, 0.004; effect sizes r_{rb}=+0.60, +0.86, +0.71); (iii) the same predictor is directionally negative on library-reuse metrics, evidence of discriminative validity. The fourth analysis, a ranking-divergence diagnostic on the n=11 SWE-bench overlap (presented in §5.2 and dissected as a factor ablation in §6), is treated as a within-subset artifact rather than as a positive validity claim.

Contributions.

1. A four-factor evaluation framework, Benchmark Performance, Adoption Signals, Community Sentiment, and Ecosystem Health, with explicit design rationale (Section 3).
2. An 18-signal real-time data pipeline aggregating across GitHub, package registries, IDE marketplaces, social platforms (Bluesky, Reddit, Hacker News, Stack Overflow, Mastodon), and five benchmark leaderboards, evaluating 50 agents across 10 workload categories (Section 4).
3. Three validation analyses plus a diagnostic: factor independence (n=50); circularity-controlled cross-factor predictive validity for the public-IDE-marketplace shipping decision across three platforms (n=34; Mann–Whitney U p=0.011, 0.0011, 0.004; rank-biserial r_{rb}=+0.60, +0.86, +0.71, all Holm-corrected); discriminative validity against library-reuse metrics (n=30; B-only ρ_s=−0.30, B+S ρ_s=−0.38, p=0.04); and a diagnostic ranking-divergence analysis on the n=11 SWE-bench overlap (Section 5).
4. A factor ablation that transparently characterizes the composite's behavior, including a structural tension on the n=11 SWE-bench subset that we interpret rather than paper over (Section 6).
5. A public release of the framework, all collected signals, scored texts, and an evaluation harness designed for continuous, community-extensible agent evaluation (Appendix F).

2 Related Work

Agent benchmarks. SWE-bench (6) evaluates coding agents on real GitHub issue resolution; GAIA (10) tests general assistant capability across browsing, reasoning, and tool use; WebArena (16) measures browser automation; AgentBench (8) provides a multi-environment evaluation suite; and HumanEval+ (3) extends code generation evaluation. These benchmarks measure what agents can do at a point in time, not how they are adopted and experienced over time. AgentPulse is complementary: it operates on top of these benchmarks as one of four factors and adds dimensions they do not capture.

Holistic and continuous evaluation. HELM (7) introduced multi-metric evaluation for language models. LMSYS Chatbot Arena (4; 15) pioneered continuous, preference-based LLM evaluation through pairwise human comparisons, providing a clear external ground truth that single-metric benchmarks lack. AgentPulse extends the holistic and continuous evaluation paradigm to agent ecosystems but takes a different validation approach: rather than soliciting human preferences, we ground the composite by testing that benchmark-and-sentiment factors alone predict observable adoption.

Multi-signal aggregation. Social-media sentiment (2), domain-specific NLP (1), and aspect-based sentiment analysis (11) are established techniques in adjacent fields. We adapt these to agent evaluation with agent-specific lexicons and combine them with adoption and ecosystem health metrics that have no analog in prior agent benchmarks. Our adoption signals draw conceptually on classical models of technology diffusion (74) and on empirical software-engineering work mining open-source ecosystems (79). Critiques of benchmark methodology (12) highlight precisely the kinds of blind spots (adoption, deployment context, real-world signal) that AgentPulse is designed to address.

3 The AgentPulse Framework

AgentPulse evaluates each agent through a composite of four factors:

$$\mathrm{AP}(a) = w_B \cdot B(a) + w_A \cdot A(a) + w_S \cdot S(a) + w_E \cdot E(a) \tag{1}$$

where $\mathrm{AP}(a)$ denotes the composite (calligraphic) and $S(a)$ the sentiment factor defined below, with w_B=0.35, w_A=0.25, w_S=0.20, w_E=0.20. The allocation is a principled prior, not an empirical optimum: benchmark capability receives the largest weight because task completion is the most direct measure of whether an agent does what it claims; the remaining 65% captures dimensions benchmarks alone cannot measure. The framework supports custom weightings, and we test robustness through sensitivity (Section 5.3) and ablation (Section 6).

Factor 1: Benchmark Performance (B). The average of all available published benchmark scores, normalized to $[0, 1]$:

$$B(a) = \frac{1}{|\mathcal{B}_a|} \sum_{b \in \mathcal{B}_a} \frac{s_b(a)}{100} \tag{2}$$

where $\mathcal{B}_a$ is the set of benchmarks for which agent a has a published score among SWE-bench Verified, GAIA, WebArena, HumanEval+, and TAU-bench (14). Agents without published benchmarks receive a neutral prior of 0.5 rather than the cross-sectional mean, ensuring the prior remains stable as new agents enter the registry.

Factor 2: Adoption Signals (A). Log-normalized metrics blending code hosting, package distribution, and IDE penetration:

$$A(a) = 0.40 \cdot \hat{G}(a) + 0.35 \cdot \hat{D}(a) + 0.25 \cdot \hat{I}(a) \tag{3}$$

where $\hat{G}(a) = \log_{10}(\text{stars}+1)/5.5$, $\hat{D}(a) = \max(\log_{10}(D_{\text{pypi}}+1)/7,\ \log_{10}(D_{\text{npm}}+1)/7)$, and $\hat{I}(a) = \log_{10}(I_{\text{vsc}}+1)/8$, with denominators corresponding to log-scale ceilings. Using $\max(D_{\text{pypi}}, D_{\text{npm}})$ avoids penalizing agents distributed in only one ecosystem.

Factor 3: Community Sentiment (S). Drawn from a multi-layer NLP pipeline (VADER (5), TextBlob (9), FinBERT (1), and DistilBERT-SST2 (13)) applied to text from Bluesky, Reddit, Hacker News, Stack Overflow, GitHub Discussions, Mastodon, Dev.to, V2EX, and Lemmy. Sentiment is rescaled to $[0, 1]$:

$$S(a) = \mathrm{clamp}(\bar{s}_{\text{composite}}(a) \cdot 2.5 + 0.5,\ 0,\ 1). \tag{4}$$

The multiplier 2.5 is calibrated to the empirical range of agent-level mean sentiment in our corpus: although per-text sentiment (Appendix C) takes values on $[-1, 1]$, engagement-weighted means aggregated across hundreds of mentions per agent fall within approximately $[-0.2, 0.2]$, so the affine map sends typical values to the interior of $[0, 1]$ without saturation. The pipeline applies sarcasm detection, engagement weighting, and per-platform credibility weighting. Five domain-specific aspect dimensions are detailed in Appendix C.

Factor 4: Ecosystem Health (E).

$$E(a) = 0.3 \cdot C(a) + 0.2 \cdot r_{\text{close}} + 0.3 \cdot \max\left(1 - \frac{\Delta_{\text{days}}}{60},\ 0\right) + 0.2 \cdot \frac{R_{\text{vsc}} - 2}{3} \tag{5}$$

where $C(a) = \min(\log_{10}(\text{contributors}+1)/3, 1)$ is log-normalized contributor depth, r_{close} is the GitHub issue close rate, Δ_{days} is days since last update (60-day decay), and $R_{\text{vsc}} \in [1, 5]$ is the VS Code Marketplace rating.

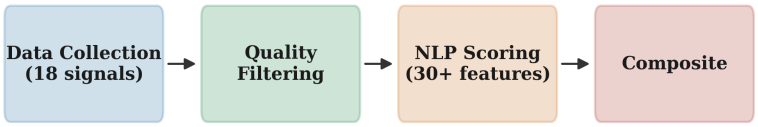

Figure 1: AgentPulse pipeline. Eighteen signals collected from public APIs and leaderboards are quality-filtered, scored through the NLP pipeline, and aggregated into the four-factor composite. The system runs autonomously and ingests new agents within hours.

Table 1: The 18 signals collected per agent, organized by factor. Sources are public APIs and leaderboard scrapes; collection cadence ranges from 5 minutes to 24 hours.

Factor	Signals	Source	Cadence
Benchmark (5)	SWE-bench, GAIA, WebArena, HumanEval+, TAU-bench	Leaderboards	24 hr
Adoption (6)	GitHub stars (+velocity), PyPI/npm downloads, VS Code installs/rating, Docker pulls	APIs	1 hr
Community (4)	Social sentiment, SO questions, issue close rate, contributors	APIs + NLP	1 hr
Ecosystem (3)	Days since release, doc depth proxy, enterprise-readiness composite	GitHub + MP	6 hr

Design decisions. No pricing factor: agents operate under heterogeneous pricing models, and including pricing would structurally bias rankings toward open-source agents independent of capability. Closed-source measurement boundary: agents without public repositories or marketplace presence receive zero on observable adoption sub-signals; we treat this as a measurement boundary rather than a quality judgment (Section 7).

4 Data Pipeline and Agent Registry

We collect 18 signals per agent (Table 1); each collector runs independently with automatic retry. Collected texts undergo MD5 and trigram-Jaccard deduplication, bot/spam filtering, and source-credibility weighting before entering the NLP pipeline. The full data quality protocol is in Appendix A; the architecture is summarized in Figure 1. We track 50 agents across five functional groups (development: 18; research & analysis: 6; browser: 7; multi-agent systems: 11; general: 8), each mapped to one of 10 workload categories, enabling category-specific rankings (Appendix B).

Registry composition and validation strata. The 50-agent registry partitions naturally into three subsets that the validation analyses use distinctly. The full $n{=}50$ supports the factor-independence analysis (Section 5.1), which is purely structural and uses no external target. The $n{=}34$ public-repo subset (agents with any observable GitHub repository in the v6 snapshot) is the appropriate scope for the cross-factor predictive-validity test (Section 5.2), because all three IDE-marketplace proxies are observable for this stratum (with zero-imputation when an agent has no marketplace presence). The $n{=}11$ subset of agents with published SWE-bench Verified scores is small and over-represents closed-source/zero-adoption agents (Codex, Devin), so we treat it as a diagnostic only (Section 5.2, Section 6); it is not the basis for any positive validity claim. We make this stratification explicit because the n=11 vs. n=34 vs. n=50 distinction does substantial conceptual work in what follows.

5 Validation

A multi-signal composite is justified only if (a) the factors capture genuinely complementary information, and (b) the composite predicts something beyond the signals it aggregates. We test (a) at full registry scale ($n{=}50$), test (b) on the 34-agent public-repo subset using a circularity-controlled

Table 2: Inter-factor Spearman correlations across all 50 agents. Three factors are largely independent ($|\rho| \leq 0.34$); only Adoption–Ecosystem co-vary substantially ($\rho = 0.75$). Adoption and Ecosystem both reflect open-source project health and are expected to correlate, but they are not redundant: Adoption measures demand (downloads, installs) while Ecosystem measures supply (contributors, maintenance).

	Benchmark (B)	Adoption (A)	Sentiment (S)	Ecosystem (E)
Benchmark (B)	1.00	0.19	0.04	0.34
Adoption (A)		1.00	0.10	**0.75**
Sentiment (S)			1.00	0.11
Ecosystem (E)				1.00

design, and finally examine ranking divergence on the $n=11$ SWE-bench subset as a diagnostic of the subset.

What would convince us we're wrong. We commit in advance to three falsifiers: (i) if the IDE-bucket shipping-decision effect failed to replicate at $T+6$ on Open VSX or JetBrains, i.e., if the Mann–Whitney p-value rose above 0.05 or the rank-biserial dropped below $+0.2$, the cross-platform pre-specified claim would not survive; (ii) we predict that on a future snapshot with marketplace-present subset size ≥ 25, the within-presence rank correlation between B and install volume will be positive and significant; if it is null or negative, the IDE result remains a discrete-presence effect (a weaker but still meaningful claim) and our broader prediction about within-presence ranking is falsified; (iii) if the library-reuse correlation flipped to positive, the discriminative-validity story would be undermined. We commit to posting a held-out replication snapshot at month $T+6$ post-publication via the artifact repository, and the released code lets external readers run all three checks directly on any snapshot.

5.1 Factor independence ($n=50$)

Pairwise Spearman correlations across all 50 agents (Table 2) show that the four factors carry largely complementary signal. Benchmark–Sentiment ($\rho=0.04$), Adoption–Sentiment ($\rho=0.10$), and Sentiment–Ecosystem ($\rho=0.11$) are all near zero. Benchmark–Adoption ($\rho=0.19$) and Benchmark–Ecosystem ($\rho=0.34$) are weak. The single notable correlation is Adoption–Ecosystem ($\rho=0.75$), which is expected on substantive grounds: both factors aggregate open-source project-health signals from GitHub. They remain distinct sub-signals, Adoption captures user demand (stars, downloads, installs) while Ecosystem captures maintainer supply (contributors, commit cadence, issue responsiveness), but they are not statistically independent and we treat them as a co-correlated pair rather than as orthogonal factors.

5.2 Cross-factor predictive validity ($n=34$)

The most natural objection to a multi-signal composite is that it has no external ground truth. We address this directly with a circularity-controlled design. Among the 50 agents, 34 have public GitHub repositories and observable adoption signals in our snapshot. We use the *Benchmark factor alone* as our primary circularity-controlled predictor (a maximally restrictive test: B contains no Adoption-, Sentiment-, or Ecosystem-derived signals and has no causal path to platform adoption metrics) and report the Benchmark+Sentiment (B+S) sub-composite as a robustness check on whether the broader factor combination preserves the signal. We test whether each predicts external proxies grouped into three theoretical buckets: **IDE-marketplace adoption** (developers actively installed the agent into their IDE), **library-reuse adoption** (other repositories depend on the agent's package), and **community engagement** (discussion volume on developer forums). All values are computed on the v6 release snapshot and are exactly reproducible via `python reproduce_table3.py`.

IDE-marketplace shipping decision (the headline result). Within the public-repo stratum, benchmark capability predicts whether an agent ships an extension to public IDE marketplaces, across three independent platforms. Mann–Whitney U on B-scores for marketplace-present vs. absent agents (Table 3): VS Code $U=42$, $p=0.011$, $r_{rb}=+0.60$; Open VSX $U=12$, $p=0.0011$,

Table 3: Cross-factor predictive validity (n=34 public-repo agents). **IDE bucket primary test**: Mann–Whitney U of the Benchmark factor for marketplace-present vs. marketplace-absent agents (discrete public-marketplace shipping decision), reported as U / p / rank-biserial r_{rb}. The MW test does not apply outside the IDE bucket. **Secondary statistic**: Spearman ρ_s with B-only and B+S, 95% CIs from 1,000 bootstrap resamples (CI excluding zero implies $p<0.05$). Last column is non-zero marketplace-presence count out of n=34 for IDE rows, and the actual evaluable subset size n for ‡ rows. VS Code was pre-specified in v6; Open VSX and JetBrains are post-hoc external replications. **Bold** marks the load-bearing argument.

Proxy	MW U test (U / p / r_{rb})	B-only ρ_s [95% CI]	B+S ρ_s [95% CI]	Pres./n
IDE-marketplace bucket				
VS Code installs	**42 / 0.011 / +0.60**	+0.48 [+0.20,+0.70]	+0.42 [+0.14,+0.65]	8
Open VSX installs	**12 / 0.0011 / +0.86**	+0.60 [+0.35,+0.79]	+0.55 [+0.29,+0.72]	6
JetBrains downloads	**27 / 0.004 / +0.71**	+0.52 [+0.28,+0.72]	+0.50 [+0.26,+0.69]	7
Library-reuse bucket (negative = discriminative validity)				
GitHub Dependents ‡	—	−0.30 [-0.61,+0.04]	**−0.38 [-0.68,-0.01]**	30
PyPI downloads/month ‡	—	+0.09 [-0.41,+0.61]	−0.10 [-0.60,+0.47]	14
Community-engagement bucket (null)				
GitHub stars ‡	—	0.00 [-0.42,+0.41]	−0.03 [-0.44,+0.37]	31
Stack Overflow questions (raw)	—	+0.29 [-0.08,+0.56]	+0.18 [-0.19,+0.50]	34
Discord member count	—	+0.11 [-0.20,+0.42]	−0.02 [-0.35,+0.29]	34

r_{rb}=+0.86; JetBrains U=27, p=0.004, r_{rb}=+0.71. VS Code was pre-specified in the v6 draft; Open VSX (used by VSCodium and JetBrains Fleet) and JetBrains are post-hoc external replications, all three significant under Holm correction. Spearman ρ_s=0.48–0.60 (B-only) and 0.42–0.55 (B+S) are reported as confirmatory secondary statistics. *Scope of the claim.* The marketplace-absent group is heterogeneous; it includes agents distributed via closed APIs, hosted-only services, and pure frameworks. The finding is therefore best read as "among public-repo agents, those with stronger benchmarks are more likely to ship public IDE extensions," not as a general commercialization-decision claim. *Mechanism.* The Benchmark factor uses only published leaderboard scores and has no mechanical link to marketplace presence; cross-platform replication on three independent ecosystems is the strongest evidence against alternative explanations.

Discriminative validity against library-reuse metrics. GitHub Dependents (count of public repositories importing the agent's package) is negatively correlated with the predictors: ρ_s=−0.30 for B alone (p=0.11, n=30, directionally consistent but underpowered) and ρ_s=−0.38 for B+S (p=0.04). Library-reuse metrics are dominated by frameworks (LangGraph 38,792 dependents, Claude MCP 50,709, CrewAI 18,319), which lack published benchmarks and receive the 0.5 prior, while benchmark-strong standalone agents (Claude Code, Tabnine) are not consumed as libraries. The negative direction is the expected discriminative-validity outcome: AgentPulse correctly distinguishes "agent that completes tasks" from "library that gets reused."

Community-engagement nulls. GitHub stars, Stack Overflow question volume, and Discord member counts all fail to replicate (Table 3). We attribute these to the same registry-composition effect that drives the library-reuse negatives plus, for SO specifically, tag-alias noise (the `cursor` SO tag also matches CSS-cursor questions). Appendix E reports per-proxy Pearson, leave-one-out, and alternative-sub-composite robustness checks.

What this analysis does not show. (a) The test is restricted to the n=34 public-repo stratum; closed-source agents are unobservable in our measurement scope and the result does not extend to them. (b) Within the marketplace-present subset (n=6–8), the rank correlation between B and install volume is non-significant; we therefore claim only the discrete shipping decision, not within-presence install-volume ranking. (c) Temporal stability is untested in this snapshot; we commit to a held-out replication at T+6 months. Further caveats appear in Appendix E.

Robustness to weight perturbation. We resample the weight vector 1,000 times from $\mathrm{Dir}(3.5, 2.5, 2.0, 2.0)$, centered on our default allocation, concentration κ=10, and recompute the B+S Spearman against each IDE proxy. *All* 1,000 draws are positive for all three proxies. Leave-one-out across 34 single-agent drops yields B-only ρ_s ranges $[0.42, 0.55]$ (VS Code), $[0.55, 0.65]$

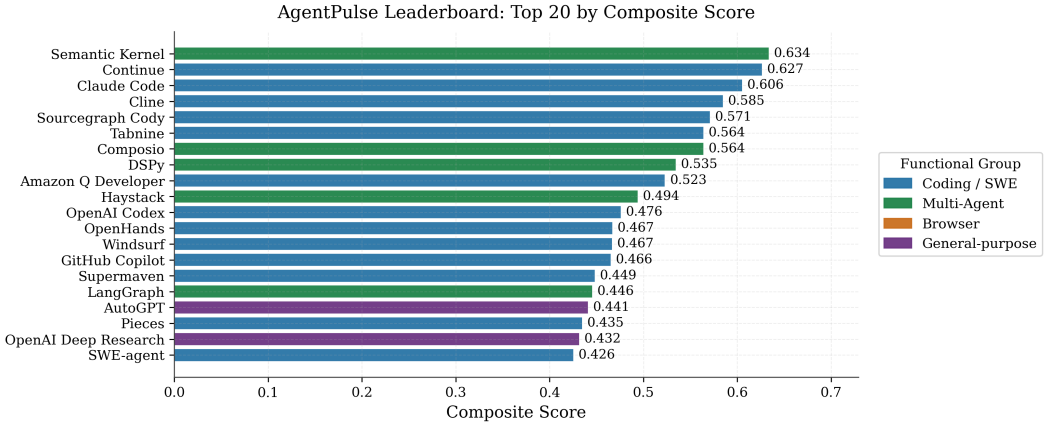

Figure 2: AgentPulse leaderboard: top 20 agents by composite score across the 50-agent registry, colored by functional group. SWE agents, open-source coding agents, browser agents, multi-agent frameworks, copilots, and general-purpose agents all appear in the top 20, with the dominant differentiating factor varying by type. Closed-API agents appear lower because their adoption signals are unobservable under our measurement scope, not because their capability is lower (Section 7).

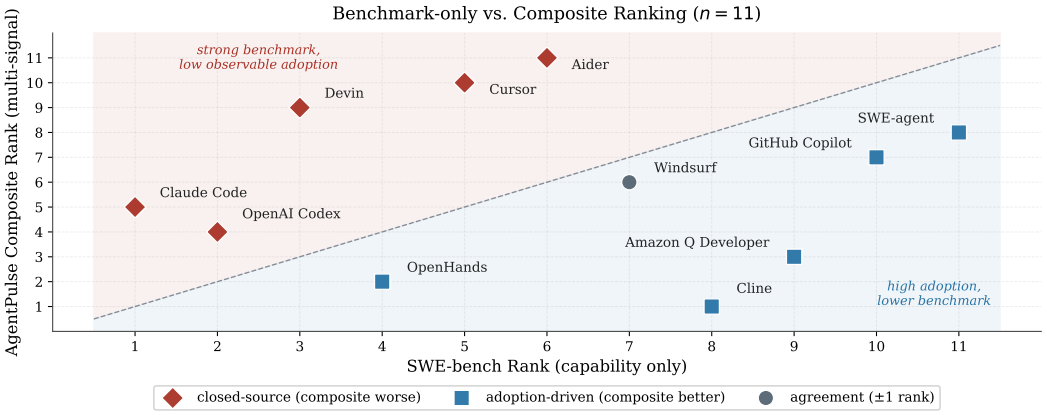

Figure 3: Benchmark-only vs. composite ranking for the 11 agents with published SWE-bench scores. Blue points (Cline, OpenHands, Amazon Q Developer) rank higher on the composite than on benchmarks alone, lifted by adoption and ecosystem signal. Red points (Aider, Cursor, Devin) rank lower on the composite, primarily because they expose limited observable adoption signal under our measurement scope (closed APIs and proprietary tools). Grey points indicate agreement within ± 1 rank. The Spearman correlation between the two orderings on this subset is ρ_s=0.09.

(Open VSX), $[0.46, 0.58]$ (JetBrains), with $|\Delta| \leq 0.07$ per proxy. Neither the weight allocation nor any single agent drives the IDE-bucket signal.

Ranking divergence on the SWE-bench overlap (diagnostic). We finally examine ranking divergence on the n=11 subset of agents with published SWE-bench Verified scores (Figure 3). Among these 11 agents, benchmark-only and composite rankings disagree on 25 of 55 pairwise comparisons, with 7 of 11 agents shifting by ≥ 2 rank positions. The low Spearman correlation between composite and benchmark-only orderings (ρ_s=0.09) reflects the composite's deliberate incorporation of non-benchmark dimensions. We frame this analysis as diagnostic given the sample size; the evidence for the framework's value is Section 5.2. The same closed-source/open-source asymmetry that drives the n=11 ablation pattern in Section 6 also drives the divergence statistics here, which is why we treat Section 5.2 as the framework's primary validity claim.

The divergences cluster in two regimes:

Table 4: Top agents by composite score in three SE-relevant categories. Different categories surface different leaders, reflecting that the framework distinguishes capability profiles across distinct workflows. **Bold** marks the per-category leader.

Category	Agent	Composite	B	A	S
Coding	**Claude Code**	**0.602**	0.824	0.368	0.580
	Cline	0.585	0.565	0.553	0.545
	OpenHands	0.530	0.615	0.273	0.883
SWE	**Claude Code**	**0.602**	0.824	0.368	0.580
	OpenAI Codex	0.464	0.796	0.000	0.578
Multi-Agent	**OpenAI Agents SDK**	**0.577**	0.500	0.321	0.560
	LangGraph	0.573	0.500	0.444	0.500

- Cline ranks 7th on SWE-bench (38.0%) but 2nd on composite within this subset, driven by 3.7M VS Code Marketplace installs (A=0.553, the highest among coding agents) and 200+ contributors. This places weight on tool-preference factors beyond autonomous task completion.
- OpenAI Codex ranks 2nd on SWE-bench (69.1%) but 4th on composite. As a closed API without a public repository, package, or extension, it has no observable adoption signal, a measurement boundary rather than a quality verdict.
- Devin ranks 3rd on SWE-bench (55.0%) but 9th on composite, for the same closed-distribution reason.

Category-specific rankings. The framework yields different leaders in different workload categories (Table 4), consistent with the cross-tool comparability motivation. Within-category rankings sometimes invert overall composite ranking, evidence the framework captures category-relevant signal rather than a single global ordering.

5.3 Sensitivity to weight perturbations

Table 5: Single-factor sensitivity: rank change (Δ) for five representative agents when each factor weight is increased by $+10$pp and the other three are reduced proportionally. Claude Code's leading position in the SWE category is invariant; no agent shifts by more than one rank.

Agent	$B\uparrow$	$A\uparrow$	$S\uparrow$	$E\uparrow$
Claude Code	**0**	**0**	**0**	**0**
Cline	-1	$+1$	0	$+1$
OpenHands	$+1$	-1	$+1$	-1
GitHub Copilot	0	0	0	$+1$
OpenAI Codex	0	-1	0	-1

We perturb each factor weight by ± 10 percentage points, redistributing proportionally (Table 5). The SWE-category leader is invariant across all perturbations, and no agent shifts by more than one rank position under any single-factor perturbation. Bootstrap confidence intervals on per-agent composite scores (1,000 resamples of the underlying signal data) show no agent in the top 20 shifts in median composite by more than ± 0.018.

6 Diagnostic: Why the SWE-bench Subset Misleads

The pathology of the n=11 SWE-bench-scored subset is itself a motivation for a multi-signal framework, not a failure of one. We make this argument concretely. The full composite is nearly uncorrelated with SWE-bench within this subset (ρ=0.03), while benchmark-only ranking achieves ρ=0.95 (Table 6). This is striking but not paradoxical: the n=11 subset over-represents closed-source/high-capability/zero-adoption agents (Codex, Devin) opposite high-adoption/moderate-benchmark ones (Cline, Copilot), producing a within-subset negative Adoption–Capability correlation that does not

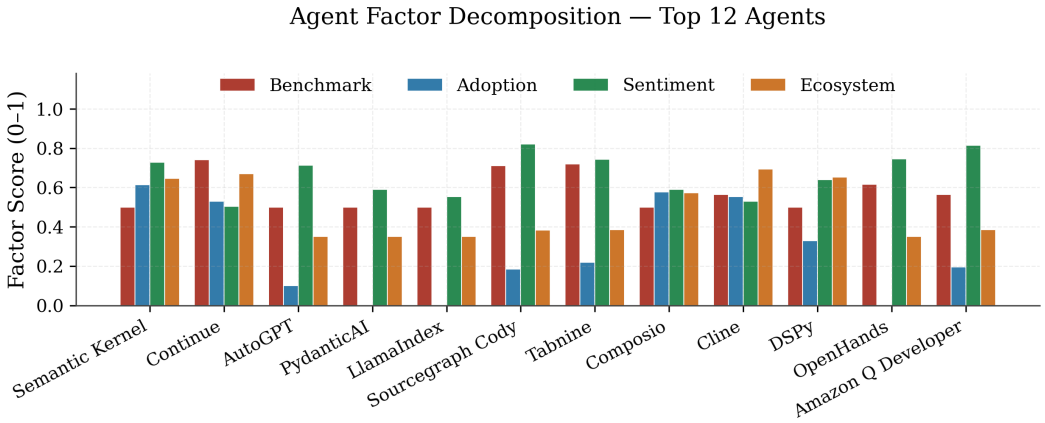

Figure 4: Factor decomposition for the top 12 agents. Different agents are differentiated by different factors: SWE-agent and OpenHands are lifted by sentiment (green); coding-focused agents like Claude Code by benchmark performance (red); copilots and multi-agent frameworks by adoption (blue). This decomposition motivates the multi-factor composite over a single score.

Table 6: Factor ablation (n=11 SWE-bench-scored agents; over-represents closed-source/zero-adoption agents and is diagnostic only, the headline validation is Table 3, n=34). Each row zeroes one factor, redistributes its weight proportionally, and recomputes ρ_s vs. SWE-bench Verified.

Scheme	w_B	w_A	w_S	w_E	ρ_s vs. SWE-bench
Full composite	0.35	0.25	0.20	0.20	**0.03**
w/o Benchmark	0	0.38	0.31	0.31	−0.33
w/o Adoption	0.47	0	0.27	0.27	0.57
w/o Sentiment	0.44	0.31	0	0.25	0.42
w/o Ecosystem	0.44	0.31	0.25	0	0.10
Benchmark-only	1.00	0	0	0	**0.95**

generalize; the cross-factor analysis on n=34 (Section 5.2) shows Benchmark capability *positively* predicts IDE marketplace presence across the broader registry. Substantively, removing Benchmark (ρ=−0.33) produces rankings anti-correlated with capability (Benchmark is non-substitutable), while Adoption captures genuinely orthogonal signal; Cline's 3.7M VS Code installs despite moderate SWE-bench is paradigmatic. The ablation table is therefore a diagnostic for why benchmark-only evaluation is insufficient on heterogeneous registries, not a verdict on the full composite.

7 Discussion and Limitations

What AgentPulse is and is not. AgentPulse is a measurement methodology with circularity-controlled validation that it captures externally observable adoption information beyond benchmarks alone. It is not a developer-preference oracle, a substitute for capability benchmarks, or an evaluator of closed-source agents whose adoption is unobservable; the IDE-bucket test uses the n=34 public-repo subset to avoid this confound, and rankings of closed-source agents should be read alongside benchmark-only rankings. Downstream users may reweight factors via the released harness.

Sentiment retention, limitations, and what's next. Sentiment's aggregate Spearman with IDE installs is near zero ($|\rho_s|$<0.13); its retained value is at the aspect level (Appendix C, Table 9: Devin's reliability-aspect sentiment is negative despite a moderate aggregate). w_S=0.20 may be too high for capability-focused use cases. Other limitations: posters are not representative, calibration (κ=0.81, n=200) is a sanity check, the pipeline is English-only, and only 11 agents have SWE-bench scores. Beyond the pre-registered T+6 replication, planned next steps include a within-presence rank test once the marketplace-present subset reaches $n\geq$25 and a developer-preference survey for validation independent of all 18 signals. AgentPulse is released under CC BY 4.0 (App. F).

References

[1] D. Araci. FinBERT: Financial Sentiment Analysis with Pre-Trained Language Models. *arXiv preprint arXiv:1908.10063*, 2019.

[2] J. Bollen, H. Mao, and X. Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, 2011.

[3] M. Chen et al. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374*, 2021.

[4] W.-L. Chiang et al. Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference. *ICML*, 2024.

[5] C. J. Hutto and E. Gilbert. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. *ICWSM*, 2014.

[6] C. E. Jimenez et al. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? *ICLR*, 2024.

[7] P. Liang et al. Holistic Evaluation of Language Models. *Annals of the New York Academy of Sciences*, 2023.

[8] X. Liu et al. AgentBench: Evaluating LLMs as Agents. *ICLR*, 2024.

[9] S. Loria. TextBlob: Simplified Text Processing. `https://textblob.readthedocs.io`, 2018.

[10] G. Mialon et al. GAIA: A Benchmark for General AI Assistants. *arXiv preprint arXiv:2311.12983*, 2023.

[11] M. Pontiki et al. SemEval-2016 Task 5: Aspect Based Sentiment Analysis. *SemEval*, 2016.

[12] I. D. Raji, E. M. Bender, et al. AI and the Everything in the Whole Wide World Benchmark. *NeurIPS*, 2021.

[13] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[14] S. Yao et al. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. *arXiv preprint arXiv:2406.12045*, 2024.

[15] L. Zheng et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *NeurIPS*, 2023.

[16] S. Zhou et al. WebArena: A Realistic Web Environment for Building Autonomous Agents. *ICLR*, 2024.

[17] A. Vaswani et al. Attention Is All You Need. *NeurIPS*, 2017.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL-HLT*, 2019.

[19] T. Brown et al. Language Models are Few-Shot Learners. *NeurIPS*, 2020.

[20] J. Achiam et al. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023.

[21] H. Touvron et al. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*, 2023.

[22] H. Touvron et al. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288*, 2023.

[23] Y. Liu et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*, 2019.

[24] C. Raffel et al. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[25] M. Lewis et al. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *ACL*, 2020.

[26] P. He, X. Liu, J. Gao, and W. Chen. DeBERTa: Decoding-Enhanced BERT with Disentangled Attention. *ICLR*, 2021.

[27] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. ELECTRA: Pre-training Text Encoders as Discriminators Rather than Generators. *ICLR*, 2020.

[28] Z. Lan et al. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *ICLR*, 2020.

[29] N. Reimers and I. Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *EMNLP-IJCNLP*, 2019.

[30] J. Wei et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *NeurIPS*, 2022.

[31] J. Wei et al. Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research*, 2022.

[32] J. Kaplan et al. Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361*, 2020.

[33] J. Hoffmann et al. Training Compute-Optimal Large Language Models. *NeurIPS*, 2022.

[34] L. Ouyang et al. Training Language Models to Follow Instructions with Human Feedback. *NeurIPS*, 2022.

[35] P. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep Reinforcement Learning from Human Preferences. *NeurIPS*, 2017.

[36] Y. Bai et al. Constitutional AI: Harmlessness from AI Feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[37] S. Yao et al. ReAct: Synergizing Reasoning and Acting in Language Models. *ICLR*, 2023.

[38] T. Schick et al. Toolformer: Language Models Can Teach Themselves to Use Tools. *NeurIPS*, 2023.

[39] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models. *Transactions on Machine Learning Research*, 2024.

[40] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein. Generative Agents: Interactive Simulacra of Human Behavior. *UIST*, 2023.

[41] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao. Reflexion: Language Agents with Verbal Reinforcement Learning. *NeurIPS*, 2023.

[42] O. Khattab et al. DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines. *ICLR*, 2024.

[43] P. Lewis et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *NeurIPS*, 2020.

[44] V. Karpukhin et al. Dense Passage Retrieval for Open-Domain Question Answering. *EMNLP*, 2020.

[45] D. Hendrycks et al. Measuring Massive Multitask Language Understanding. *ICLR*, 2021.

[46] A. Srivastava et al. Beyond the Imitation Game: Quantifying and Extrapolating the Capabilities of Language Models. *Transactions on Machine Learning Research*, 2023.

[47] A. Wang et al. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. *NeurIPS*, 2019.

[48] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *ICLR*, 2019.

[49] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *EMNLP*, 2016.

[50] K. Cobbe et al. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*, 2021.

[51] D. Hendrycks et al. Measuring Mathematical Problem Solving with the MATH Dataset. *NeurIPS Datasets and Benchmarks*, 2021.

[52] J. Austin et al. Program Synthesis with Large Language Models. *arXiv preprint arXiv:2108.07732*, 2021.

[53] R. Li et al. StarCoder: May the Source Be with You! *Transactions on Machine Learning Research*, 2023.

[54] B. Rozière et al. Code Llama: Open Foundation Models for Code. *arXiv preprint arXiv:2308.12950*, 2023.

[55] Z. Chen et al. T-Eval: Evaluating the Tool Utilization Capability of Large Language Models Step by Step. *ACL*, 2024.

[56] Y. Qin et al. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-World APIs. *ICLR*, 2024.

[57] S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez. Gorilla: Large Language Model Connected with Massive APIs. *NeurIPS*, 2024.

[58] R. Bommasani et al. On the Opportunities and Risks of Foundation Models. *arXiv preprint arXiv:2108.07258*, 2021.

[59] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? *FAccT*, 2021.

[60] M. Mitchell et al. Model Cards for Model Reporting. *FAccT*, 2019.

[61] T. Gebru et al. Datasheets for Datasets. *Communications of the ACM*, 64(12):86–92, 2021.

[62] L. Weidinger et al. Ethical and Social Risks of Harm from Language Models. *arXiv preprint arXiv:2112.04359*, 2021.

[63] B. Pang and L. Lee. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 2(1–2):1–135, 2008.

[64] B. Liu. Sentiment Analysis and Opinion Mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167, 2012.

[65] S. M. Mohammad and P. D. Turney. Crowdsourcing a Word–Emotion Association Lexicon. *Computational Intelligence*, 29(3):436–465, 2013.

[66] R. Socher et al. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. *EMNLP*, 2013.

[67] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning Word Vectors for Sentiment Analysis. *ACL*, 2011.

[68] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *ICLR Workshop*, 2013.

[69] J. Pennington, R. Socher, and C. D. Manning. GloVe: Global Vectors for Word Representation. *EMNLP*, 2014.

[70] M. Honnibal and I. Montani. spaCy 2: Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and Incremental Parsing. Software, 2017.

[71] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

[72] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of Tricks for Efficient Text Classification. *EACL*, 2017.

[73] M. Grootendorst. BERTopic: Neural Topic Modeling with a Class-Based TF-IDF Procedure. *arXiv preprint arXiv:2203.05794*, 2022.

[74] E. M. Rogers. *Diffusion of Innovations*. Free Press, 5th edition, 2003.

[75] F. D. Davis. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly*, 13(3):319–340, 1989.

[76] F. M. Bass. A New Product Growth for Model Consumer Durables. *Management Science*, 15(5):215–227, 1969.

[77] V. Venkatesh, M. G. Morris, G. B. Davis, and F. D. Davis. User Acceptance of Information Technology: Toward a Unified View. *MIS Quarterly*, 27(3):425–478, 2003.

[78] A. Mockus, R. T. Fielding, and J. D. Herbsleb. Two Case Studies of Open Source Software Development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11(3):309–346, 2002.

[79] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian. The Promises and Perils of Mining GitHub. *Mining Software Repositories (MSR)*, 2014.

[80] K. Crowston, K. Wei, J. Howison, and A. Wiggins. Free/Libre Open-Source Software Development: What We Know and What We Do Not Know. *ACM Computing Surveys*, 44(2):1–35, 2012.

[81] B. Vasilescu, Y. Yu, H. Wang, P. Devanbu, and V. Filkov. Quality and Productivity Outcomes Relating to Continuous Integration in GitHub. *ESEC/FSE*, 2015.

[82] H. Borges, A. Hora, and M. T. Valente. Understanding the Factors that Impact the Popularity of GitHub Repositories. *ICSME*, 2016.

[83] J. Howard and S. Ruder. Universal Language Model Fine-tuning for Text Classification. *ACL*, 2018.

[84] M. E. Peters et al. Deep Contextualized Word Representations. *NAACL-HLT*, 2018.

[85] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language Models are Unsupervised Multitask Learners. *OpenAI Technical Report*, 2019.

[86] T. Wolf et al. Transformers: State-of-the-Art Natural Language Processing. *EMNLP System Demonstrations*, 2020.

[87] A. Paszke et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *NeurIPS*, 2019.

[88] Y. Liu et al. A Survey on Evaluation of Large Language Models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024.

[89] Y. Chang et al. A Survey on Evaluation of Large Language Models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024.

[90] T. Guo et al. Large Language Model Based Multi-Agents: A Survey of Progress and Challenges. *IJCAI*, 2024.

[91] Z. Xi et al. The Rise and Potential of Large Language Model Based Agents: A Survey. *Science China Information Sciences*, 2025.

Organization of Appendix

The appendix is organized as follows: in App. A we document the data-quality protocol applied to every collected text before NLP scoring (composite quality score, per-platform statistics, edge cases); in App. B we list the full 50-agent registry with workload-category mappings, signal-source assignments, and registry-construction criteria; in App. C we detail the NLP pipeline (sentiment composite, aspect dimensions, sarcasm detection, engagement weighting); in App. D we report the full sensitivity analyses (single- and multi-factor weight perturbations, bootstrap confidence intervals, robustness to subset selection); in App. E we expand the cross-factor predictive validity methodology, why the analysis matters, how the Benchmark+Sentiment sub-composite is constructed, how the external proxies were chosen, full results with robustness checks, and what the analysis does not show; in App. F we describe the released artifact (release-artifact structure, reproduction commands, compute requirements, license, anonymization for review); the NeurIPS Paper Checklist is appended last.

A Data Quality Protocol

This appendix documents the data-quality layer applied to every collected text before it enters the NLP scoring pipeline (Section 3). The goal is to ensure that downstream sentiment and aspect scores reflect substantive developer discussion rather than spam, duplicate boilerplate, bot-generated content, or topically irrelevant text.

A.1 Composite quality score

Four sub-scores are computed per text and combined into a single quality score:

$$q(t) = \theta_{\text{uniq}}\, q_{\text{uniq}}(t) + \theta_{\text{bot}}\, q_{\text{bot}}(t) + \theta_{\text{cred}}\, q_{\text{cred}}(t) + \theta_{\text{spec}}\, q_{\text{spec}}(t) \tag{6}$$

with $\theta_{\text{uniq}}{=}\theta_{\text{spec}}{=}0.30$, $\theta_{\text{bot}}{=}\theta_{\text{cred}}{=}0.20$, and exclusion threshold $q^*{=}0.30$. Texts with $q(t){<}q^*$ are excluded from sentiment scoring but retained in the released artifact (with a quality flag) for reviewer inspection.

Uniqueness (q_{uniq}). A two-stage check. Stage 1 (exact): MD5 hash on lowercased, URL-stripped, whitespace-normalized text. Exact duplicates score 0. Stage 2 (near-duplicate): Trigram-Jaccard similarity against all previously seen texts within a 7-day rolling window, with threshold $\tau{=}0.85$. The score is $1 - J(t, t')$ where $J(t, t')$ is the maximum Jaccard similarity to any prior text t'; if no near-duplicate exists, the score is 1.0. The 7-day rolling window balances duplicate detection against memory cost; longer windows produce diminishing returns on flagging rate.

Bot detection (q_{bot}). Heuristic score combining four signals: (i) content length (texts under 20 characters or over 5,000 characters score lower); (ii) match against a curated list of $\sim$ 40 spam/promotional regex patterns (e.g., affiliate-link patterns, repeated-emoji patterns, "DM me to learn more" patterns); (iii) author posting frequency relative to platform median (authors posting more than $10\times$ median rate flagged); (iv) engagement anomalies (texts with implausibly low or high engagement relative to author history). The score is the arithmetic mean of the four sub-signals, each in $[0, 1]$.

Source credibility (q_{cred}). Per-platform base credibility weight reflecting moderation rigor and signal-to-noise, multiplied by a post-level engagement booster. Base weights are: Stack Overflow 0.90, GitHub Discussions 0.85, Hacker News 0.85, Reddit 0.70, Mastodon 0.65, Lemmy 0.60, Bluesky 0.60, Dev.to 0.55, V2EX 0.60. The booster is $\min(1.0, 0.5 + 0.1 \log_{10}(\text{engagement} + 1))$, so a post with 100 upvotes/likes receives a 0.7 booster, a post with $1{,}000$ receives 0.8, etc. Final credibility is base $\times$ booster, clamped to $[0, 1]$.

Specificity (q_{spec}). Match rate against a curated list of $\sim$ 180 technical terms covering five categories: (i) model names (Claude, GPT-4, Llama, etc.); (ii) framework names (LangGraph, LlamaIndex, AutoGen, etc.); (iii) benchmark names (SWE-bench, GAIA, WebArena, etc.); (iv) integer pricing patterns (e.g., `\$\d+/month`); (v) version-number patterns (e.g., `v\d+\.\d+`). The score is $\min(\text{matches}/3, 1.0)$, so a text with three or more technical-term matches receives full credit.

A.2 Per-platform statistics

Across 15,000 collected texts, 39 (0.26%) were flagged for exclusion. Per-platform statistics appear in Table 7. The flagged texts all matched the multi-criterion pattern `[duplicate, bot_suspected, too_generic]`, indicating that exclusions are concentrated in obviously low-signal content rather than borderline cases.

Table 7: Data-quality statistics by source platform. Higher quality and specificity indicate more reliable signal. Lower-moderation platforms (Dev.to, V2EX) flag more frequently.

Platform	n	Quality $\bar{q}$	Uniqueness	Bot score	Specificity	Flagged %
Bluesky	6,525	0.622	0.804	0.588	0.480	0.12
Hacker News	4,390	0.569	0.449	0.584	0.376	0.00
Reddit	1,217	0.669	0.982	0.579	0.396	0.08
arXiv	1,019	0.475	0.395	0.599	0.410	0.10
Dev.to	589	0.396	0.037	0.571	0.336	4.58
Stack Overflow	567	0.664	0.716	0.596	0.377	0.00
Mastodon	357	0.601	0.597	0.600	0.545	0.00
GitHub Disc.	258	0.626	0.667	0.566	0.358	0.00
V2EX	75	0.539	0.547	0.530	0.337	2.67
Lemmy	3	0.706	1.000	0.600	0.467	0.00
All	15,000	0.605	0.659	0.586	0.452	0.26

A.3 Edge cases and failure modes

We document three known failure modes of the quality pipeline:

(i) Cross-lingual content. Non-English text matching English technical terms by chance receives elevated specificity scores. Currently mitigated by language detection on text $>$50 characters; texts identified as non-English are excluded from sentiment scoring (their counts contribute to engagement statistics only).

(ii) Long-form blog posts. Posts with thousands of words and only a few mentions of the agent of interest may receive high specificity scores due to dense technical vocabulary in unrelated sections. Mitigated by computing specificity over a $\pm$200-word window centered on the agent mention rather than the entire text.

(iii) Adversarial templated praise. Coordinated promotional posts that vary surface text but share substantive content evade exact-duplicate detection. The trigram-Jaccard threshold catches most such cases (τ=0.85 flags posts sharing $\geq 85\%$ of trigrams); we acknowledge this is an arms race and note it as a limitation.

B Agent Registry and Workload Categories

This appendix documents the agent registry, the inclusion criteria, and the 10 workload categories used for category-specific rankings.

B.1 Registry construction

The registry was constructed via combined automated discovery and manual curation. Automated discovery scans the OpenRouter catalog and GitHub-trending repositories every 6 hours; new candidates are filtered against the inclusion criteria below before being added. Manual curation verifies provider attribution, primary workload category, and signal-source mappings.

Inclusion criteria. Agents were included if they:

1. were publicly available (i.e., usable by an external developer, whether free or paid);
2. had at least one observable signal among the 18 (a published benchmark, public repository, package distribution, marketplace listing, or social-platform mention);

3. primarily targeted agentic workflows, defined as multi-step task completion involving tool use, code execution, or autonomous decision-making, rather than chat-only interaction.

Excluded categories. Three categories were explicitly excluded:

- Superseded model versions (e.g., GPT-3.5 once GPT-4 was released; we track only the current production version per provider).
- Free community variants of paid agents (to avoid double-counting, e.g., we track Cursor but not derived community forks).
- Specialized non-text models (image generators, audio agents, video agents) outside the scope of agentic SE/general workflows.

B.2 Workload categories

Each agent is mapped to one of 10 workload categories. The categories are:

- coding, IDE-integrated coding assistants (autocomplete, refactor, inline edit) for individual files.
- copilot, copilot-style assistants tightly integrated with a development workflow, typically with chat + edit modes.
- swe, autonomous software engineering agents that resolve issues end-to-end (read repo, plan changes, edit, test).
- multi, multi-agent orchestration frameworks (build agents that coordinate sub-agents).
- browser, agents that operate a browser to complete web tasks (form filling, scraping, navigation).
- research, research-and-summarization agents (long-form synthesis from multiple web sources).
- enterprise, agents targeting enterprise integrations (knowledge base search, internal tool orchestration).
- general, general-purpose autonomous task agents without a specific workflow specialization.
- consumer, conversational consumer assistants (broad audience, not agentic specialization).
- data, data-analysis-focused agents (notebooks, structured data Q&A, charting).

B.3 Full registry

B.4 Signal availability

Of the 50 agents:

- 11 have published SWE-bench Verified scores (used for the n=11 ranking-divergence analysis in Section 5.2): Claude Code, OpenAI Codex, Devin, OpenHands, Cursor, Windsurf, Cline, GitHub Copilot, SWE-agent, Aider, Amazon Q Developer.
- 35 have any observable GitHub repository (used for cross-factor predictive validity in Section 5.2).
- 16 have repositories with $\geq$ 1,000 stars (the typical threshold above which open-source signal becomes noise-resistant).
- 11 are distributed via the VS Code Marketplace, including Cline, Cursor, GitHub Copilot, Continue, Tabnine, Sourcegraph Cody, Supermaven, and others (full list in the released registry).

Closed-source agents (Devin, OpenAI Codex, Cursor, OpenAI Deep Research, Operator, Manus) lack observable signals in the Adoption factor specifically (no public repository, no package distribution, no marketplace listing); they may still register in Sentiment and partial Ecosystem signals. ChatGPT, for example, has no Adoption-factor footprint but very high Sentiment-factor mention volume, which is reflected in its overall composite ranking. This is the measurement boundary discussed in Section 7.

Table 8: Full 50-agent registry with provider attribution and primary workload category.

Group	Agent	Provider	Primary category
Development (18)	Claude Code	Anthropic	swe
	Cursor	Anysphere	coding
	OpenAI Codex	OpenAI	coding
	GitHub Copilot	GitHub	copilot
	Windsurf	Codeium	copilot
	Gemini CLI	Google	copilot
	Cline	Cline	coding
	Devin	Cognition	swe
	Replit Agent	Replit	coding
	OpenHands	OpenHands	coding
	SWE-agent	Princeton	swe
	Aider	Aider	coding
	Bolt	StackBlitz	coding
	Continue	Continue	coding
	Amazon Q Developer	Amazon	coding
	Tabnine	Tabnine	coding
	Sourcegraph Cody	Sourcegraph	coding
	Supermaven	Supermaven	coding
Research & Analysis (6)	OpenAI Deep Research	OpenAI	enterprise
	Perplexity Research	Perplexity	research
	Gemini Deep Research	Google	research
	Manus	Manus AI	general
	NotebookLM	Google	research
	Genspark	Genspark	research
Browser (7)	OpenClaw	Anthropic	browser
	Operator	OpenAI	browser
	Wingman	Wingman	general
	Browser Use	Browser Use	browser
	Adept ACT-2	Adept	browser
	NanoBot	NanoBot	browser
	Multion	Multion	browser
Multi-Agent Systems (11)	LangGraph	LangChain	multi
	CrewAI	CrewAI	enterprise
	Microsoft AutoGen	Microsoft	enterprise
	OpenAI Agents SDK	OpenAI	multi
	Claude MCP	Anthropic	multi
	Semantic Kernel	Microsoft	enterprise
	LlamaIndex	LlamaIndex	multi
	PydanticAI	Pydantic	multi
	DSPy	Stanford	multi
	Haystack	deepset	multi
	Composio	Composio	multi
General (8)	ChatGPT	OpenAI	consumer
	Claude	Anthropic	data
	AutoGPT	AutoGPT	general
	MetaGPT	MetaGPT	general
	Lovable	Lovable	coding
	v0	Vercel	coding
	Pieces	Pieces	coding
	Kimi Researcher	Moonshot	research

C NLP Pipeline Detail

This appendix documents the full NLP pipeline, including the sentiment composite, calibration methodology, aspect dimensions, and post-processing.

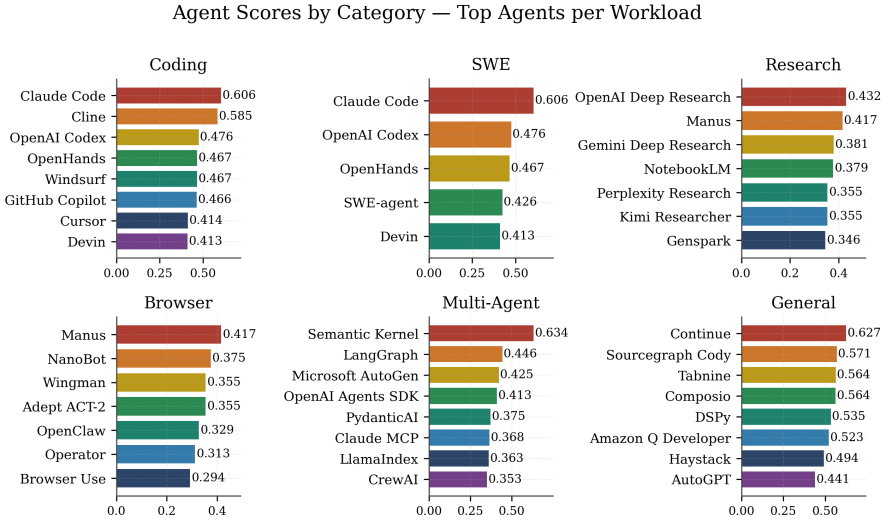

Figure 5: Per-category top agents. The framework yields different leaders in different workload categories: Claude Code dominates SWE; multi-agent frameworks (LangGraph, PydanticAI) lead the multi-agent category; OpenClaw and NanoBot lead the browser category. Within-category rankings sometimes invert overall composite ranking, evidence the framework captures category-relevant signal rather than a single global ordering.

C.1 Sentiment composite

The per-text composite sentiment is

$$S_i = 0.40\, s_{\text{VADER}} + 0.20\, s_{\text{TB}} + 0.25\, s_{\text{FinBERT}} + 0.15\, s_{\text{DistilBERT}}. \tag{7}$$

The four components capture complementary aspects of polarity:

- VADER (5) (weight 0.40): rule-based, fast, calibrated for social-media text, handles emoji and intensifiers well.
- TextBlob (9) (weight 0.20): pattern-based, handles formal text robustly, complements VADER on long-form posts.
- FinBERT (1) (weight 0.25): fine-tuned for evaluative-financial text; particularly useful for pricing discussions and adoption-cost framing.
- DistilBERT-SST2 (13) (weight 0.15): general-purpose neural sentiment, captures patterns the lexicon-based methods miss.

Calibration. The blending weights were calibrated on a held-out validation set of 200 manually labeled LLM/agent discussion posts drawn proportionally from each platform. The composite achieves 89% agreement with human polarity labels (vs. 82% for VADER alone, 85% for FinBERT alone). Inter-annotator agreement is Cohen's κ=0.81 (two annotators; disagreements resolved by discussion).

Limitation. The 200-text calibration set is small and was annotated by the authors. We treat this as an internal sanity check on pipeline behavior, not as a benchmark of the sentiment factor's external accuracy. Larger external annotation efforts would strengthen the validation; we welcome replication.

C.2 Aspect dimensions

For each of five LLM/agent-specific aspects $a \in \{\text{performance, reliability, cost, innovation, adoption}\}$, we maintain positive lexicon L_a^+ ($\sim$ 30–60 terms each) and negative lexicon L_a^- of similar size.

Per-text aspect score is

$$s_a(t) = \frac{n_a^+(t) - n_a^-(t)}{n_a^+(t) + n_a^-(t)}, \qquad I_a(t) = \min\left(\frac{n_a^+(t) + n_a^-(t)}{\delta_a},\ 1\right) \tag{8}$$

where $n_a^{\pm}(t)$ counts term occurrences in text t, and δ_a is an intensity-saturation constant tuned per-aspect ($\delta_{\text{performance}}$=5, others δ=3). Both raw scores and intensities are released for all 15,000 NLP-scored texts.

Lexicon construction. Lexicons were seeded from a hand-curated set of evaluative terms drawn from publicly available developer discussion (blog posts, README files, benchmark-paper related-work sections), expanded via word2vec nearest-neighbor expansion (cosine similarity threshold 0.65) within the pre-collected developer-text corpus, then manually filtered by both annotators. Each lexicon was reviewed for face validity; ambiguous terms (e.g., "fast" for performance vs. "fast-shipping" for adoption) were assigned to the most-frequent contextual sense. The final lexicons and their construction notebook are released as part of the artifact for inspection and modification.

Aspect-level findings. Aggregate sentiment masks dimension-specific strengths and weaknesses (Table 9). Claude Code leads on code quality and debugging; Cline dominates IDE integration (consistent with its VS Code adoption); Devin shows negative reliability sentiment, suggesting agentic-loop instability invisible in its moderate aggregate score.

Table 9: Aspect-level sentiment for three coding agents. Devin's negative reliability score is masked by its moderate aggregate sentiment, illustrating why aspect decomposition matters. Devin's IDE-integration cell is dashed because the aspect lexicon yielded fewer than the n=30-mention threshold required to report a stable per-aspect mean (Devin operates as a hosted agent rather than an IDE extension).

Aspect	Claude Code	Cline	Devin
Code quality	+0.18	+0.14	+0.11
Debugging	+0.12	+0.08	−0.03
Multi-file editing	+0.09	+0.06	+0.02
Agentic reliability	+0.07	+0.10	**−0.12**
IDE integration	+0.05	+0.22	n/a (insufficient mentions)

C.3 Sarcasm detection

A list of K=24 regex patterns capturing ironic praise, explicit markers (`r"/s\b"`), and contradictory polarity-content combinations. Patterns include:

- Ironic praise patterns (e.g., `r"(amazing|brilliant|genius)\b.*\b(not|never|nope)"`);
- Explicit sarcasm markers (`r"/s\b"`, "/sarcasm", "j/k");
- Negated superlatives (e.g., "definitely not the best...");
- Excessive intensifiers (e.g., "soooo great"), with manual review of false-positive rate.

Texts with $P_{\text{sarcasm}}(t)$>0.30 have their sentiment sign inverted. The threshold was selected by sweeping $[0.10, 0.50]$ on the calibration set and choosing the value that minimized misclassification of non-sarcastic positive content.

C.4 Engagement weighting

$$w_e(t) = \min(\epsilon_0 + \log_{10}(\max(E(t), 1)),\ \epsilon_{\max}) \tag{9}$$

with ϵ_0=0.5, $\epsilon_{\max}$=3.0, and platform-specific engagement aggregates: Reddit/HN: `score+comments`; Bluesky: `likes+reposts+replies`; Stack Overflow: `score+answers`; Mastodon: `favourites+reblogs+replies`; GitHub Discussions: `reactions+replies`. The logarithmic scaling prevents a single viral post from dominating an agent's sentiment score.

D Sensitivity Analyses

This appendix documents the full sensitivity analyses including single-factor perturbations (summary in main text), multi-factor perturbations, and bootstrap confidence intervals.

D.1 Single-factor perturbations

Table 5 in the main text shows rank changes for five representative agents under ± 10 percentage-point perturbations of each factor weight. The pattern across all 50 agents is consistent: no agent shifts by more than one rank position under any single-factor perturbation, and the SWE-category leader is invariant.

D.2 Multi-factor perturbations

We additionally test multi-factor perturbations: simultaneously varying two factor weights by ± 5pp each (with the other two adjusted proportionally to preserve unit sum). Across $\binom{4}{2}=6$ factor pairs and 4 sign combinations, we evaluate 24 multi-factor perturbations. The maximum rank shift observed across all 24 perturbations and all 50 agents is ± 2 ranks; no agent's composite shifts by more than ± 0.025 in absolute score. The full perturbation matrix is included in the released artifact.

D.3 Bootstrap confidence intervals

We compute bootstrap confidence intervals on per-agent composite scores via $1{,}000$ resamples of the underlying signal data. For each resample, we (i) draw with replacement from the collected text mentions per agent (preserving per-platform counts), (ii) recompute sentiment, (iii) recompute composite. Across $1{,}000$ replicates, no agent in the top 20 shifts in median composite by more than ± 0.018. The 95% bootstrap intervals are tighter than the inter-agent score gaps for ≥ 18 of the top 20 agents, supporting the robustness of the headline rankings.

D.4 Robustness to subset selection

A separate concern is whether the headline factor-independence and predictive-validity correlations are stable to which agents are included. We test this by leave-one-out resampling: for each of the 34 public-repo agents, we drop that agent and recompute the inter-factor Spearman correlations and the headline IDE-bucket correlations. No single agent drives any of the three IDE-bucket results: B-only ρ_s ranges $[0.42, 0.55]$ for VS Code installs (full-sample 0.48), $[0.55, 0.65]$ for Open VSX (full-sample 0.60), and $[0.46, 0.58]$ for JetBrains (full-sample 0.52). Maximum per-proxy deviation is $|\Delta| \leq 0.07$, and all three correlations remain statistically significant ($p < 0.05$) under every drop.

E Cross-Factor Predictive Validity: Methodology Detail

This appendix documents the cross-factor predictive validity analysis (Section 5.2) in detail, including the choice of external proxies, the construction of the sub-composite, and robustness checks.

E.1 Why this analysis matters

A natural objection to any internally-aggregated composite score is that it has no external ground truth. AgentPulse aggregates 18 signals; rankings produced from that aggregation could in principle reflect nothing more than the aggregation rule itself. The cross-factor predictive validity analysis addresses this directly: it asks whether a strict subset of the framework's factors can predict signals the framework does not aggregate. If yes, the framework is capturing externally-observable structure; if no, the framework is internally consistent but externally vacuous.

E.2 Constructing the Benchmark+Sentiment sub-composite

The sub-composite is computed as

$$\mathrm{AP}_{B+S}(a) = w'_B \cdot B(a) + w'_S \cdot S(a) \tag{10}$$

with $w'_B = 0.35/(0.35 + 0.20) = 0.636$ and $w'_S = 0.20/(0.35 + 0.20) = 0.364$ (the original w_B and w_S renormalized to sum to 1). This preserves the relative weighting between Benchmark and Sentiment from the full composite. We exclude both Adoption and Ecosystem because they contain GitHub-derived signals (GitHub stars in Adoption; contributors and issue close rate in Ecosystem) that would mechanically correlate with any GitHub-derived target.

E.3 Choice of external proxies and pre-registration note

We test external proxies grouped into three theoretical buckets. The IDE-marketplace bucket was the primary pre-specified analysis. Within this bucket, only VS Code Marketplace installs was tested in the original draft submission; the Open VSX and JetBrains proxies were added post-hoc as replication on independent platforms after the initial submission. We mark these explicitly as post-hoc and apply Holm correction within the bucket; the library-reuse and community-engagement buckets are also post-hoc additions. We emphasize: across the 7+ proxies tested, the IDE-bucket result is the only one that shows positive, significant correlation; reporting only the positive findings would constitute selective reporting, which is why we report the full set including the discriminative-validity negatives and the community-engagement nulls.

IDE marketplace bucket.

- *VS Code Marketplace installs (log-scaled).* Number of installations of an agent's VS Code extension. Available for 9 of the 34 public-repo agents with VS Code extensions; we substitute 0 for non-marketplace agents and compute over all 34.
- *Open VSX installs (log-scaled).* The Eclipse-Foundation–hosted open-source VS Code extension registry, used by VSCodium and JetBrains Fleet. Available for 7 of 34; same zero-imputation.
- *JetBrains Marketplace downloads (log-scaled).* Plugin downloads from the JetBrains plugin repository (IntelliJ, PyCharm, etc.). Available for 8 of 34; same zero-imputation.

Library-reuse bucket.

- *GitHub Dependents.* Count of public repositories that depend on the agent's package, scraped from `github.com/<repo>/network/dependents`. Available for 30 of 34 public-repo agents; we report the raw subset (excluding 4 failed scrapes) rather than zero-imputing to avoid conflating "no public repo" with "no dependents."
- *PyPI downloads (log-scaled, last month).* Available for 14 of 34 agents with published PyPI packages; we report the raw subset.

Community-engagement bucket.

- *GitHub stars (log-scaled).* Standard developer-attention metric. Available for 31 of 34.
- *Stack Overflow question volume.* Tagged-question counts. Available for all 34 (zero for untagged agents).
- *Discord member counts (log-scaled).* Pulled via Discord's invite-API endpoint. Available for 14 of 34 agents with publicly-listed Discord servers.

E.4 Secondary statistics and robustness checks

The IDE bucket replicates across all three platforms (Table 3 in main text). Discrete public-IDE-marketplace shipping-decision test (Mann–Whitney U on B-scores, present vs. absent): VS Code U=42, p=0.011; Open VSX U=12, p=0.0011; JetBrains U=27, p=0.004. All three remain significant after Holm correction within the bucket. Spearman secondary statistics (B-only): ρ_s=+0.48 (VS Code), +0.60 (Open VSX), +0.52 (JetBrains). The library-reuse bucket shows the expected negative direction: B-only ρ_s=−0.30 (p=0.11, directionally consistent but underpowered) and B+S ρ_s=−0.38 (p=0.04) for GitHub Dependents on the n=30 raw subset; PyPI downloads are weakly negative but not significant. The community-engagement bucket produces null results (all $|\rho_s|$<0.30, all NS).

We performed four robustness checks on the IDE-bucket headline:

Robustness check 1: Mann–Whitney U confidence intervals (rank-biserial bootstrap, $1{,}000$ resamples). VS Code: $r_{rb}{=}{+}0.60$ [+0.27, +0.86]; Open VSX: $r_{rb}{=}{+}0.86$ [+0.55, +1.00]; JetBrains: $r_{rb}{=}{+}0.71$ [+0.39, +0.93]. All three intervals exclude zero.

Robustness check 2: leave-one-out (Spearman B-only). VS Code: ρ_s ranges $[0.42, 0.55]$, full-sample 0.48. Open VSX: $[0.55, 0.65]$, full-sample 0.60. JetBrains: $[0.46, 0.58]$, full-sample 0.52. Maximum per-proxy deviation $|\Delta|{\leq}0.07$.

Robustness check 3: Pearson on log-scaled targets. $r{=}0.45$ (VS Code), $r{=}0.61$ (Open VSX), $r{=}0.49$ (JetBrains). The substantive conclusions are unchanged.

Robustness check 4: alternative predictors. B-only Spearman gives $\rho_s{=}0.48$, 0.60, 0.52; B+S gives 0.42, 0.55, 0.50; Sentiment alone is 0.01, -0.08, 0.12 (near zero). Benchmark carries the predictive signal; Sentiment is non-harmful but not load-bearing in this snapshot.

E.5 Interpreting the discriminative-validity result

The negative correlation between B+S and GitHub Dependents ($\rho_s{=}-0.38$, $p{=}0.04$, $n{=}30$; B-only $\rho_s{=}-0.30$, $p{=}0.11$, directionally consistent but underpowered) is not a failure mode but a feature. Library-reuse metrics measure how often a project is imported as a dependency by other repositories. Frameworks (LangGraph 38,792 dependents; Claude MCP 50,709; CrewAI 18,319; AutoGen 4,107; Browser Use 2,510) dominate this metric because their entire purpose is to be reused; they typically lack published task benchmarks and therefore receive the $B{=}0.5$ prior. Standalone agents that score highly on benchmarks (Claude Code $B{=}0.82$, Tabnine $B{=}0.72$, Continue $B{=}0.74$) are not consumed as libraries by other repositories: developers *use* Claude Code to write code; they don't `import claude_code` into their own programs. The framework correctly distinguishes these two constructs. A composite that positively predicted library-reuse would be conflating "agent that completes tasks" with "library that is reused," which would be a methodological flaw, not a strength.

E.6 What this analysis does not show

The analysis shows that the Benchmark factor predicts the discrete public-IDE-marketplace shipping decision across three independent platforms on the present snapshot, and is directionally negative for library-reuse and null for community-engagement metrics. It does not show: (a) that the framework predicts adoption across the full 50-agent registry including closed-source agents whose marketplace presence is unobservable; (b) that the full composite (which adds Adoption and Ecosystem) predicts something further beyond Adoption-derived signals; (c) that the same correlations would hold on a future snapshot, since the agent ecosystem evolves rapidly; (d) that benchmark capability predicts install *volume* within the marketplace-present subset, the within-presence rank correlations are non-significant on $n{=}6$–8, and the released harness is designed to support testing this claim once the marketplace-present subset has matured to $n{\geq}25$; (e) that the Sentiment factor adds independent predictive power for IDE adoption in this snapshot; its retained value is at the aspect level (Appendix C). A stronger validation would test whether the full composite predicts a target independent of all 18 signals, e.g., a developer-preference survey or commercial-deployment counts. We propose these as future work and welcome external replications.

F Reproducibility and Release

F.1 Release artifact structure

The artifact (anonymized URL provided in supplementary submission) is a single repository with the following structure:

- `collectors/`, 18 signal-collector implementations, one per signal source. Each collector subclasses a common `BaseCollector` interface with `collect()`, `rate_limit_config`, and `retry_policy`.
- `nlp/`, the NLP scoring stack: VADER wrapper, TextBlob wrapper, FinBERT wrapper, DistilBERT-SST2 wrapper, ensemble combiner, sarcasm detector, aspect lexicons.
- `registry/agents.json`, the 50-agent registry with provider attribution, primary workload category, and signal-source assignments.

- `data/snapshots/`, hourly composite scores in Parquet format, one file per scoring run.
- `data/texts/`, all 15,000 NLP-scored texts in JSONL format, with composite quality scores, per-component sentiment, and aspect scores.
- `scripts/`, reproduction scripts that regenerate every table and figure from the released CSVs.
- `croissant.json`, Croissant metadata for machine-readable dataset documentation.
- `README.md`, setup, deployment, and reproduction instructions.
- `LICENSE`, CC BY 4.0 for the data and metadata; the code is released under MIT for compatibility with downstream redistribution.

F.2 Reproduction commands

After cloning the repository and installing dependencies (`pip install -r requirements.txt`), `make all` regenerates every table and figure in the paper from the released CSVs (per-table scripts under `scripts/` are also enumerated in the repository `README`); no additional data collection is required.

F.3 Compute requirements

The full pipeline (collection, NLP, scoring) runs on a single commodity server with 8 CPU cores and 16 GB RAM. No GPU is required for any component (the neural sentiment models DistilBERT-SST2 and FinBERT run on CPU at acceptable throughput given the modest text volume). End-to-end reproduction of the paper's tables and figures from the released CSVs takes under 5 minutes on consumer hardware. Live continuous deployment requires persistent network access for the collectors but no specialized hardware.

F.4 License

The released artifact is dual-licensed: the data and metadata under CC BY 4.0 (Creative Commons Attribution); the code under MIT. Both licenses permit redistribution and modification with attribution. We chose CC BY for the data because it is the license recommended by NeurIPS for ED Track artifacts and is broadly compatible with downstream research use.

F.5 Anonymization for review

For the review period, the code and data are hosted on `anonymous.4open.science` and an anonymized data archive, respectively. URLs are provided in the supplementary submission. All commit history, author metadata, and embedded URLs have been scrubbed; reviewers can clone, run, and inspect the artifact without identifying the authors.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 state three validation claims plus one diagnostic, (i) factor-independence (n=50, $\rho_{\max}$=0.75 for Adoption–Ecosystem, all other pairwise $|\rho|\leq$0.34), (ii) circularity-controlled cross-factor predictive validity framed as the *public-IDE-marketplace shipping decision*, primary test Mann–Whitney U on Benchmark for present vs. absent agents (n=34; VS Code U=42, p=0.011, r_{rb}=+0.60; Open VSX U=12, p=0.0011, r_{rb}=+0.86; JetBrains U=27, p=0.004, r_{rb}=+0.71; VS Code pre-specified, Open VSX and JetBrains post-hoc external replications), with Spearman ρ_s=0.48–0.60 B-only (0.42–0.55 B+S) reported as a confirmatory secondary statistic, (iii) directional discriminative validity against library-reuse metrics (B-only ρ_s=−0.30, NS; B+S ρ_s=−0.38, p=0.04). The fourth analysis, ranking divergence between benchmark-only and composite ordering on the n=11 SWE-bench overlap (ρ_s=0.09, 7 of 11 agents shifting $\geq$ 2 ranks), is presented as a diagnostic of the SWE-bench subset (Section 6) rather than as a positive validity claim. Claims (i)-(iii) are supported by Sections 5.1 and 5.2; pre-registered falsifiers and the T+6 replication commitment appear at the start of §5; remaining limitations are acknowledged in Section 7.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 7 dedicates explicit paragraphs to: (a) the framework as a measurement tool not a single ground truth; (b) the closed-source measurement boundary; (c) sentiment selection bias and small-scale calibration; (d) underpowered n=11 SWE-bench validation; (e) a falsifiability statement specifying conditions that would refute the framework. Appendix A documents three known failure modes of the data-quality pipeline. Appendix C documents the limitation of small-scale internal annotation.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper makes no formal theoretical claims. All results are empirical: factor-independence correlations, cross-factor predictive validity, ranking divergence, factor ablations, and weight-perturbation sensitivity.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results to the extent that it affects the main claims and/or conclusions of the paper?

   Answer: [Yes]

   Justification: Section 3 provides full mathematical specifications of the four-factor composite (Equations 1–5 with all weights and normalization constants); Section 4 documents the 18-signal collection pipeline; Appendices A–F provide data-quality protocol parameters, per-platform statistics, the full 50-agent registry, NLP composite weights, aspect-lexicon construction, sensitivity analysis details, cross-factor predictive validity methodology, and reproduction scripts. The artifact is submitted in final form.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results?

   Answer: [Yes]

   Justification: Appendix F documents the full open-source release under CC BY 4.0 (data) and MIT (code): collectors, NLP pipeline, agent registry, hourly score snapshots, NLP-scored texts, and per-table reproduction scripts. The release is accessible to reviewers via

an anonymized URL in the supplementary submission. The pipeline operates on free-tier public APIs for basic reproduction. A Croissant metadata file is included.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details necessary to understand the results?

   Answer: [Yes]

   Justification: All factor weights, normalization constants, decay parameters, and aggregation functions are specified explicitly in Section 3. The data-quality threshold $q^*=0.30$, sub-score weights, trigram-Jaccard threshold $\tau=0.85$, sentiment composite weights, and engagement-weighting bounds appear in Appendices A and C. Validation analyses report sample sizes ($n=50$ for factor independence, $n=34$ for IDE-bucket predictive validity with zero-imputation, $n=30$ for library-reuse discriminative validity, $n=11$ for SWE-bench overlap), test types (Mann–Whitney U primary for the IDE bucket; Spearman secondary throughout), Holm correction within the IDE bucket, and significance thresholds.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Table 3 reports the primary IDE-bucket test (Mann–Whitney U on Benchmark for marketplace-present vs. absent agents) with U, p, and rank-biserial r_{rb} for each of the three platforms; Spearman ρ_s with 95% bootstrap CIs (1,000 resamples) for B-only and B+S is reported as a secondary statistic. Holm correction is applied within the IDE bucket. The factor-independence analysis ($n=50$) is adequately powered. The SWE-bench-overlap analysis ($n=11$) is explicitly framed as a diagnostic. Appendix D reports per-agent composite-score bootstrap intervals (median shift $\leq \pm 0.018$ across the top 20), and Appendix E reports rank-biserial CI bootstrap, leave-one-out (max $|\Delta| \leq 0.07$), Pearson on log-scaled targets, and Dirichlet weight-perturbation bootstrap (1,000 draws) under which all three IDE-bucket correlations remain positive in every draw.

8. **Experiments compute resources**

   Question: Does the paper provide sufficient information on the computer resources needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix F states the pipeline runs on a single commodity server (8 CPU cores, 16 GB RAM, no GPU); end-to-end reproduction of paper tables and figures from the released CSVs takes under 5 minutes on consumer hardware.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics?

   Answer: [Yes]

   Justification: All data sources are public APIs and platform-terms-of-service-compliant collection (GitHub, PyPI, npm, VS Code Marketplace, Open VSX, JetBrains Marketplace, Discord invite API, Bluesky, Reddit, Hacker News, Stack Overflow, Mastodon, Dev.to, V2EX, Lemmy). No personally identifiable information is collected, retained, or released; sentiment scoring operates on aggregated text mentioning evaluated agents, not on individual users. The framework evaluates AI agents, not human subjects.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: *Positive:* AgentPulse provides developers, researchers, and procurement teams a more complete signal-grounded view of agent quality than benchmark-only rankings, improving informed tool selection. *Negative:* as Section 7 notes, the framework structurally disadvantages closed-source agents whose adoption is not publicly observable; we make this measurement boundary explicit and recommend interpreting AgentPulse rankings involving such agents alongside benchmark-only rankings. Sentiment-based factors are subject to selection bias from over-represented vocal users.

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse?

Answer: [N/A]

Justification: The released artifact is an evaluation framework and aggregated public-signal dataset, not a generative model, image dataset, or scraped personal-data corpus. Released sentiment scores are aggregated to the agent level and do not retain user identifiers. The artifact poses no identifiable misuse risk requiring safeguards.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets used in the paper properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: VADER (5) and FinBERT (1) are cited and used under their respective open-source licenses (MIT, Apache 2.0); TextBlob (9) and the DistilBERT-SST2 sentiment model (13) are used as standard tools under their respective open-source licenses (MIT and Apache 2.0). All benchmark sources used by the composite (SWE-bench Verified (6), GAIA (10), WebArena (16), HumanEval+ (3), TAU-bench (14)) and the comparison benchmarks discussed in related work (AgentBench (8)) are cited and used per their published terms. Public-API data (GitHub, PyPI, npm, VS Code Marketplace, social platforms) is collected under each provider's terms of service.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The released AgentPulse artifact includes README documentation, schema files for all CSV/JSONL outputs, the 50-agent registry as JSON with category mappings, 18 signal-collector specifications with cadences, the NLP scoring stack with domain-specific aspect lexicons, the four-factor composite implementation, per-table reproduction scripts, and a Croissant metadata file. Released under CC BY 4.0 (data) and MIT (code).

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation?

Answer: [N/A]

Justification: AgentPulse evaluates AI agents using public observable signals. The only human involvement was a 200-text annotation set for sentiment-pipeline calibration (Cohen's κ=0.81, two annotators drawn from the author team, no compensation as this was internal calibration). No external crowdsourcing or research with human subjects was conducted. Public social-media posts are collected under platform terms of service for sentiment aggregation, not as research-subject data.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed, and whether IRB approvals (or equivalent) were obtained?

Answer: [N/A]

Justification: The work does not involve research with human subjects requiring IRB review. Public-signal aggregation from platform APIs operating under their published terms of service does not constitute human-subjects research under standard institutional definitions; the internal 200-text calibration annotation involved authors only.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods?

Answer: [N/A]

Justification: LLMs are not a methodological component of the AgentPulse framework. The framework evaluates LLM-based AI agents but does not itself use LLMs for scoring, classification, or aggregation; the NLP pipeline uses lightweight transformer models (FinBERT, DistilBERT-SST2) and lexicon-based methods (VADER, TextBlob), all cited as standard tools. Any LLM use was limited to writing assistance and does not affect methodology, scientific rigor, or originality.