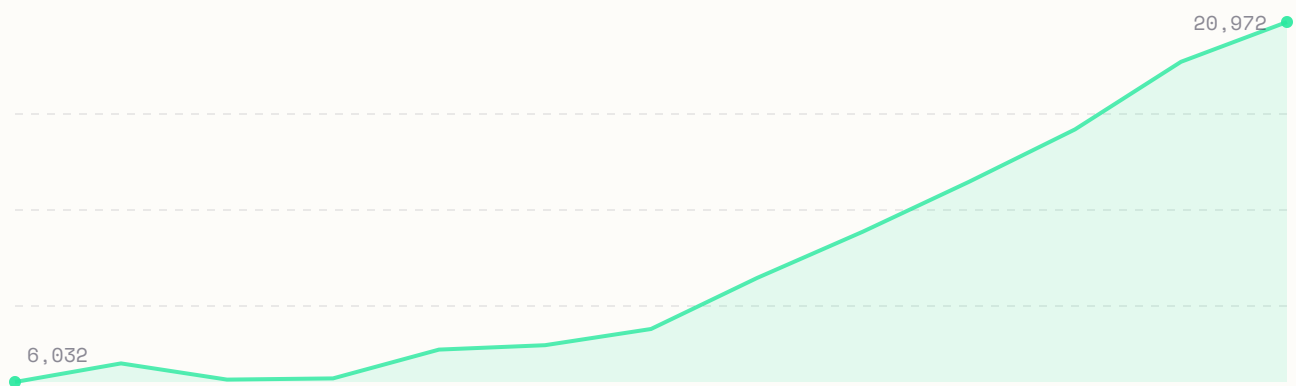


Q2 2026 UPDATE

A CROSS-INDUSTRY STUDY ON AI SOFTWARE DEVELOPMENT

The State of AI Coding



LINES OF CODE PER DEVELOPER • JUN 2025 - JUN 2026

+79%

MEDIAN PR SIZE, YOY

3.5x

LINES PER DEVELOPER

80%

ORGS USING CLAUDE.MD

Table of Contents

	01 Engineering Team Velocity Measuring productivity gains across development workflows	PAGE 03
	02 AI Tool Adoption Tracking the rise of AI-powered development tools	PAGE 05
	03 Model Growth Trends How AI models have scaled and evolved	PAGE 07
	04 Research & Content Recent papers on foundational models and applications	PAGE 08
ABOUT THIS REPORT		PAGE 13

METHODOLOGY & SOURCES

- Engineering velocity (Section 01): Greptile internal merge-request dataset over a fixed March 2025 repository cohort; medians computed over merged and closed pull requests.
- Tool adoption and model trends (Sections 02 and 03): PyPI and npm monthly download counts; where noted, automated uv-installer and CI traffic is excluded.
- Rules-file adoption: presence of CLAUDE.md, AGENTS.md, and Cursor rules files across Greptile-connected organizations that use an AI rules file.

All figures as of June 2026 unless noted. Published July 2026.

</>

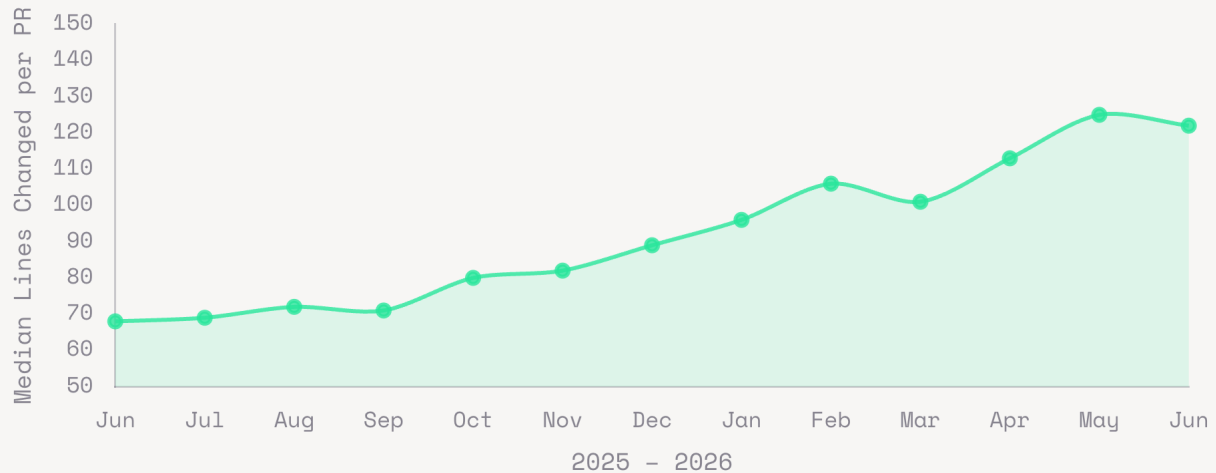
Engineering Team Velocity

</>

Measuring productivity gains across development workflows.

CHART 1.1 PRs Are Getting Bigger

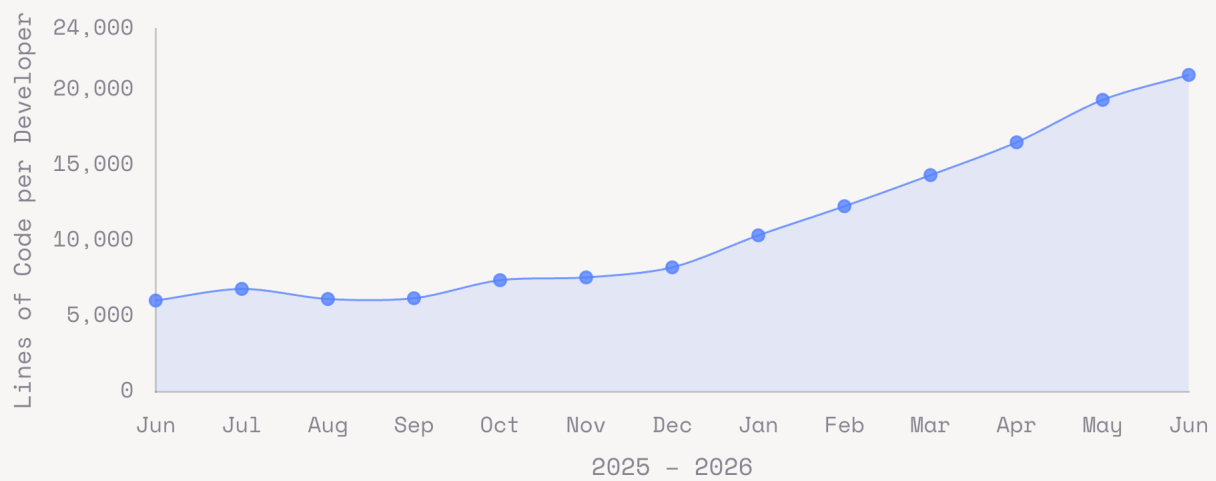
Median PR size grew 79% from June 2025 to June 2026, rising from 68 to 122 lines changed per PR.



Captured from Greptile internal data engineering team velocity

CHART 1.2 Developer Output

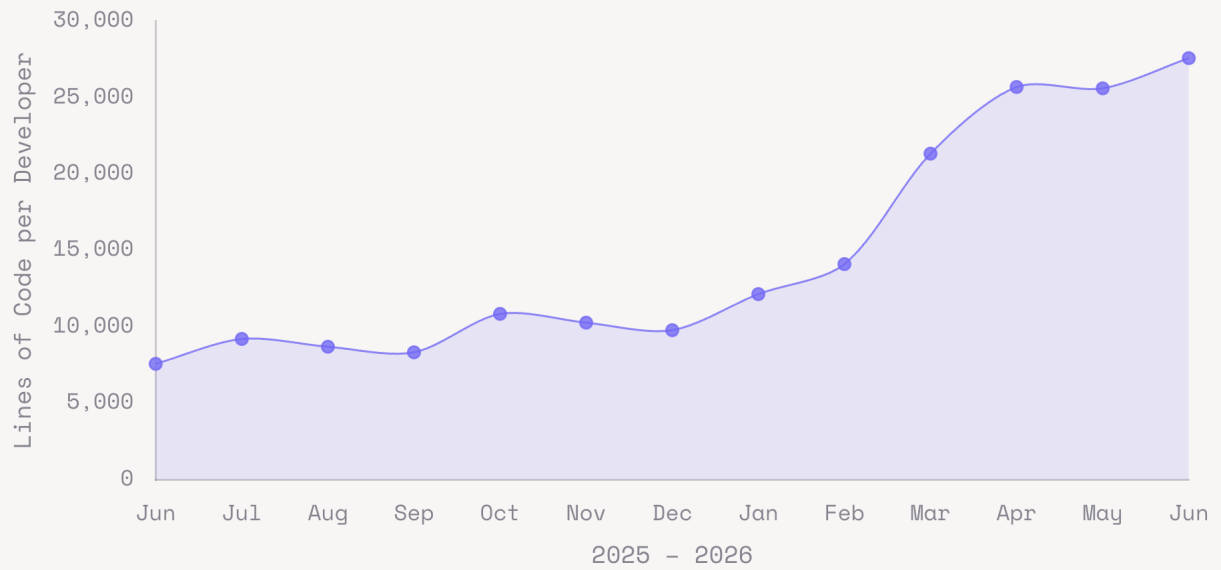
Lines of code per developer grew 3.5x over the past year, from 6,032 to 20,972, a steady climb as AI coding tools act as a force multiplier.



Captured from Greptile internal data engineering team velocity

CHART 1.3 Medium Teams Output

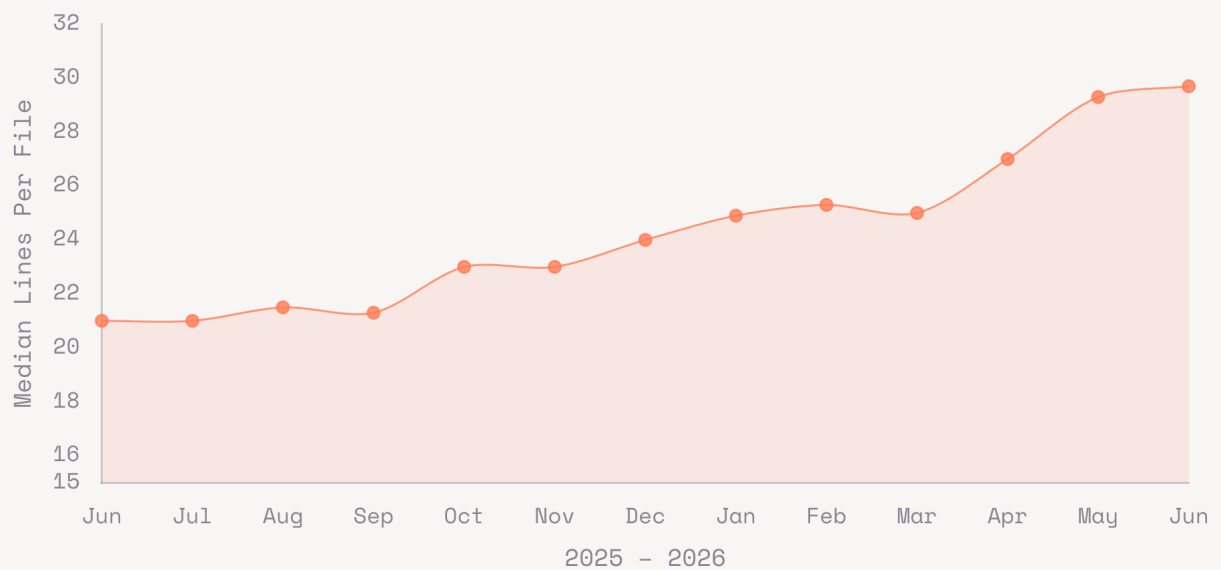
Medium teams (6–50 devs) more than tripled output, from 7,576 to 27,584 lines per developer.



Captured from Greptile internal data engineering team velocity

CHART 1.4 Lines Changed Per File

Median lines changed per file grew from 21 to 30 as PRs become denser.



Captured from Greptile internal data engineering team velocity



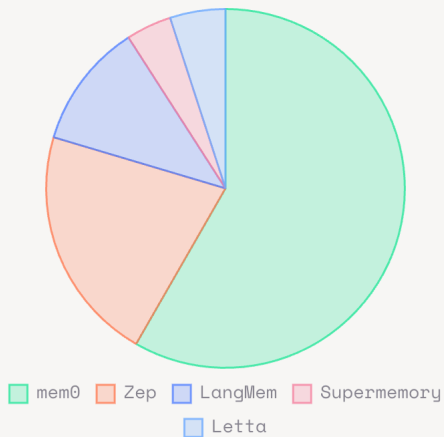
AI Tool Adoption



Tracking the rise of AI-powered development tools.

CHART 2.1 AI Memory Packages

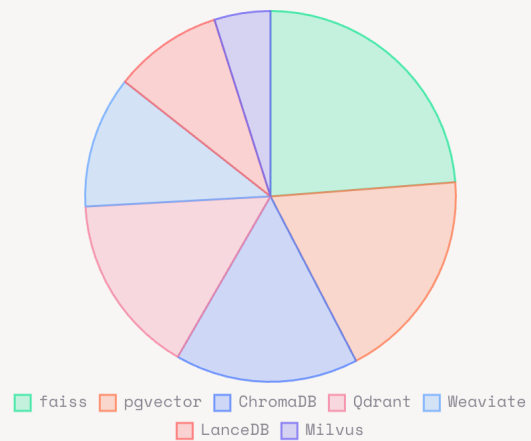
mem0 holds 58% market share. LangMem slipped while Letta and Supermemory gained ground.



PyPI + npm monthly downloads, May 2026

CHART 2.2 Vector DB Developer Installs

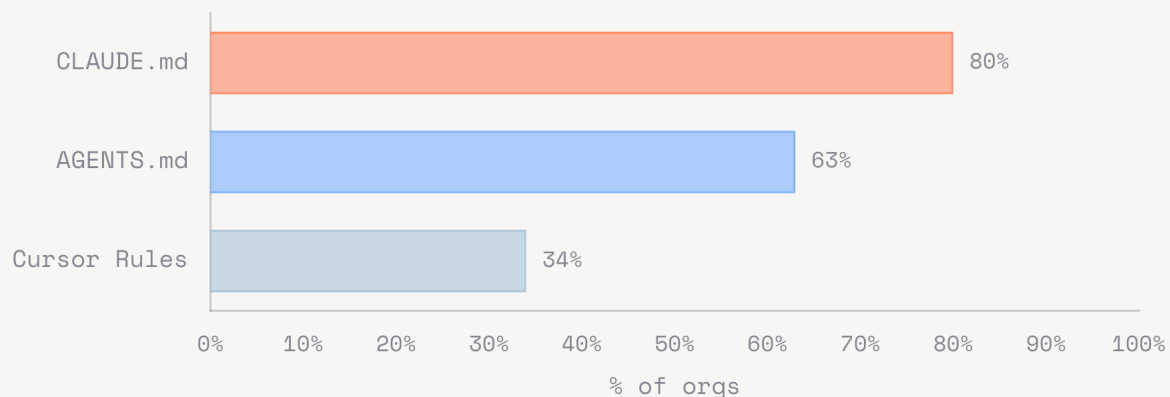
faiss and pgvector lead on developer installs. Weaviate sits mid-pack at 11% after excluding automated uv-installer traffic.



Open-source vector DBs, PyPI pip installs, June 2026. uv-installer traffic excluded.

CHART 2.3 AI Rules Files

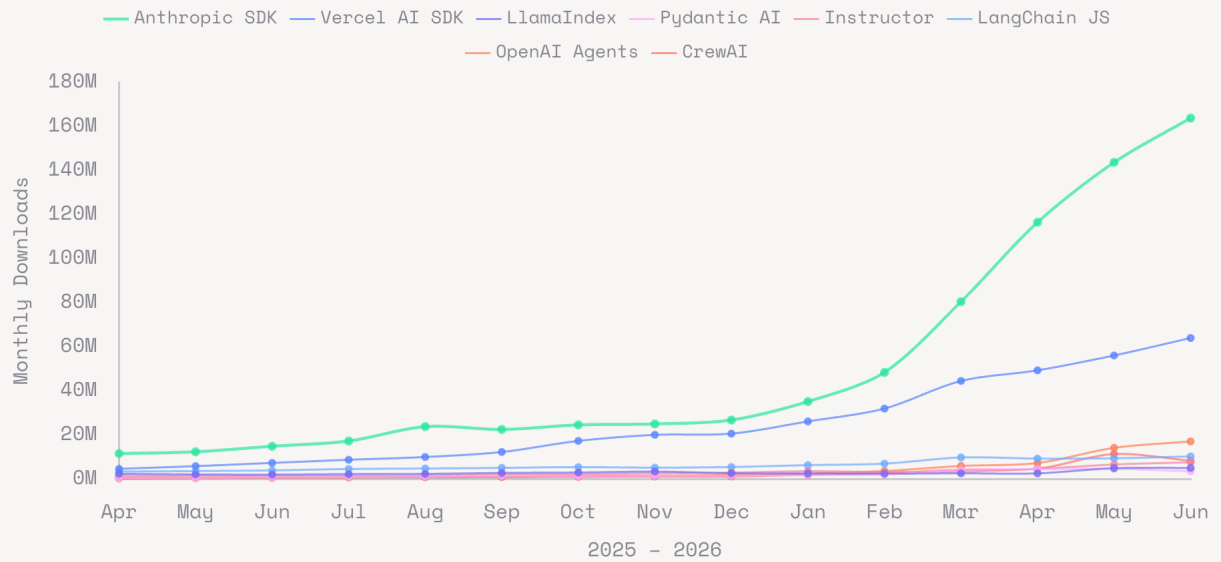
CLAUDE.md present in 80% of Greptile orgs that use an AI rules file. AGENTS.md climbed to 63%, while Cursor Rules slid to 34%.



About 18% of orgs use all three formats, roughly flat since December (17%), even as CLAUDE.md and AGENTS.md adoption climbed

CHART 2.4 AI SDK Growth

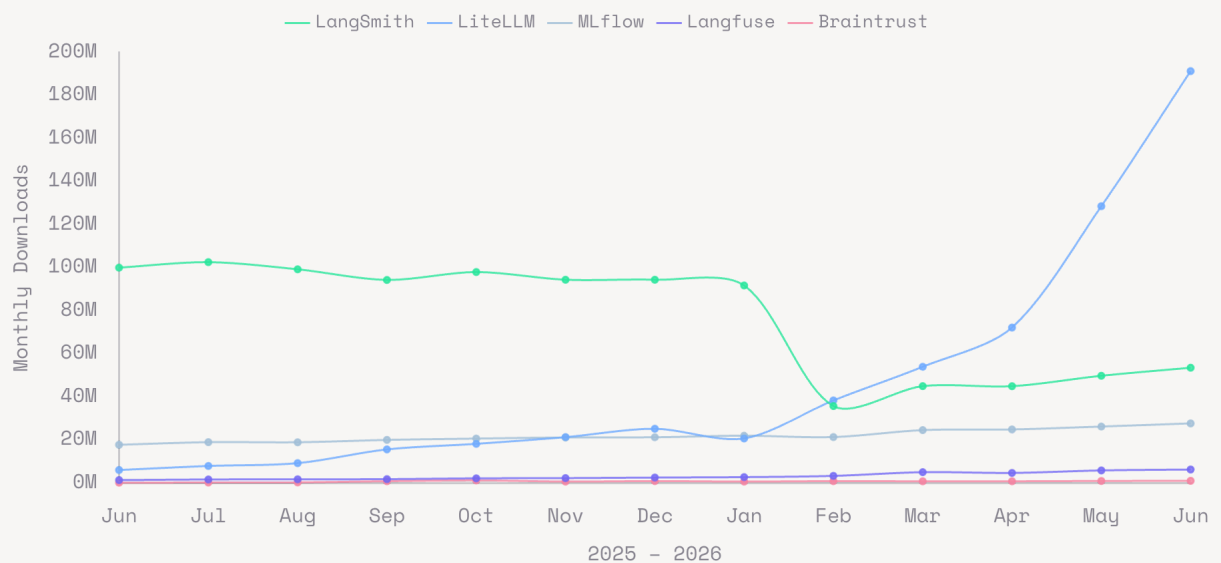
Anthropic SDK reached 164M monthly downloads by June 2026 (npm + pip). OpenAI Agents climbed to 17M.



PyPI (pip installer) + npm monthly downloads, Apr 2025 - Jun 2026

CHART 2.5 LLMOps Top 5

LiteLLM's pip installs overtook LangSmith, reaching 191M/month by June 2026.



PyPI downloads via pip installer (uv/CI traffic excluded), Jun 2025 - Jun 2026. LangSmith is bundled with LangChain installs.



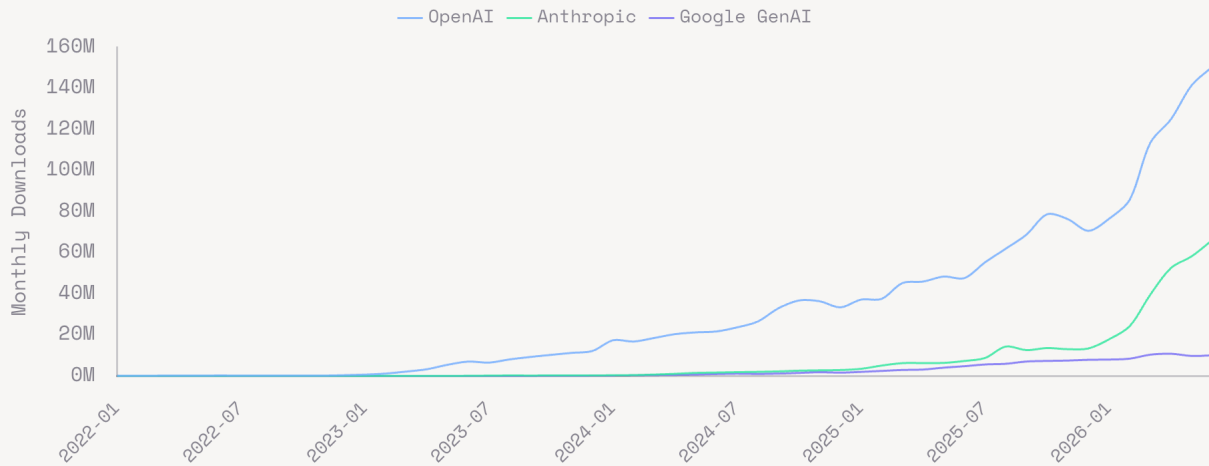
Model Growth Trends



How AI models have scaled and evolved.

CHART 3.1 LLM Provider SDK Downloads

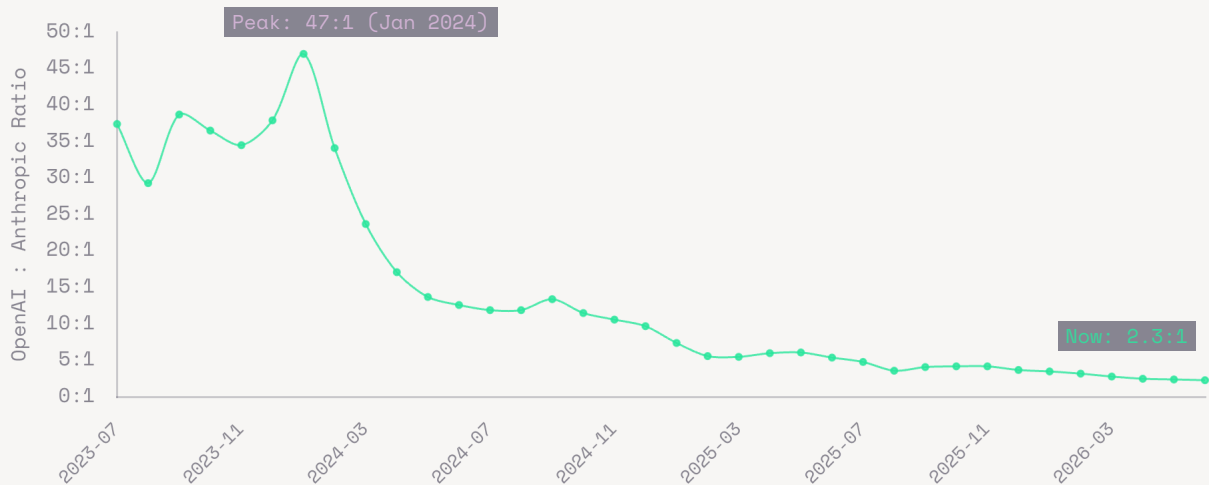
OpenAI reached 150M monthly pip installs by June 2026. Anthropic surged to 66M. Google (legacy SDK) trails at ~10M.



PyPI downloads via pip installer (uv/CI traffic excluded), Jan 2022 – Jun 2026

CHART 3.2 The Gap is Closing

OpenAI-to-Anthropic ratio kept narrowing, from 3.7:1 (Dec 2025) to 2.3:1 (June 2026).



PyPI monthly downloads ratio, Jul 2023 – Jun 2026



Research & Content



Surfacing recent research that shaped how 2026 tools handle compression, context, multimodality, and long-horizon agents.

Foundational Model Advances

PAPERS 1 - 6

TurboQuant: Online Vector Quantization with Near-optimal Distortion Rate

arxiv.org/abs/2504.19874 ↗

TurboQuant is an online, data-oblivious quantization method for KV-cache compression and vector search that targets both mean-squared error and inner-product distortion.

- It randomly rotates vectors so coordinates are easier to quantize, then applies scalar quantizers coordinate-wise instead of relying on slower vector quantization schemes.
- To correct the inner-product bias of MSE-optimal quantization, it adds a 1-bit Quantized Johnson-Lindenstrauss step on the residual; the paper reports quality-neutral KV-cache compression at 3.5 bits per channel and stronger recall than product-quantization baselines in vector search.
- Long-context inference and retrieval depend on compression schemes that preserve the geometry attention and nearest-neighbor search actually use, not just on storing vectors in fewer bits.

Recursive Language Models

arxiv.org/abs/2512.24601 ↗

RLMs treat the prompt as external state, letting the model inspect, partition, and recursively call itself over snippets instead of feeding everything through one context window.

- The model works inside a Python REPL where the prompt is stored as a variable, letting it inspect slices of the input, execute code, and recursively call itself over selected snippets.
- Across CodeQA, BrowseComp-Plus, and OOLONG, RLMs can handle inputs up to two orders of magnitude beyond base context windows and outperform direct model calls and retrieval-style scaffolds.
- Long-context scaling may come less from ever-larger windows and more from giving models programmable ways to search, decompose, and recurse over external state.

Titans: Learning to Memorize at Test Time

arxiv.org/abs/2501.00663

Titans is a family of long-context architectures that pairs limited-window attention with a neural long-term memory that keeps learning at test time.

- Its memory module writes surprising inputs into its weights using gradient-based updates, then uses momentum and weight decay as a learned forgetting mechanism.
- The paper instantiates the design as Memory as Context (MAC), Memory as Gating (MAG), and Memory as Layer (MAL); across language, reasoning, needle-in-a-haystack, time series, and genomics tasks, the authors report stronger results than recent recurrent and hybrid baselines, with MAC doing best on longer dependencies.
- Overall, long-context modeling may benefit more from explicit write/forget/retrieve memory systems than from relying on ever-larger attention windows alone.

Kimi K2.5: Visual Agentic Intelligence

arxiv.org/abs/2602.02276

Kimi K2.5 is an open-source multimodal agentic model that jointly optimizes text and vision, with a parallel-agent execution framework layered on top.

- It mixes vision tokens early during pretraining, uses a native-resolution MoonViT-3D encoder for images and video, and follows with zero-vision SFT plus joint multimodal RL.
- Agent Swarm uses a trainable orchestrator to spawn frozen specialist subagents and run heterogeneous subtasks concurrently; the paper reports 76.8% on SWE-Bench Verified and large gains on BrowseComp with 3–4.5× lower execution time at target quality.
- Agentic performance may come less from a single monolithic model doing everything in sequence and more from jointly trained multimodal foundations paired with learned orchestration over parallel specialist work.

VeriCache: Turning Lossy KV Cache into Lossless LLM Inference

arxiv.org/abs/2605.17613

VeriCache is an inference framework that keeps the throughput of compressed KV caches while producing the exact outputs of full-KV decoding.

- It uses the compressed KV cache to draft tokens and then verifies them against the full KV cache, which is held outside GPU memory and swapped in on demand so that verification overhead stays low.
- The paper reports up to 4x higher throughput than full-KV inference while producing identical outputs, targeting the divergence that makes lossy compression risky for code generation and tool calling as more tokens are decoded.
- For long-context coding agents, the useful frontier for KV compression is not just a smaller memory footprint but behavior that stays correct across long generations, tool calls, and code paths.

M2A: Synergizing Mathematical and Agentic Reasoning in Large Language Models

arxiv.org/abs/2605.09879 ↗

M2A is a model-merging method for adding mathematical reasoning to a coding agent without disrupting its agentic behavior.

- The paper argues math reasoning (closed-world, single-response logic) and agentic reasoning (multi-turn interaction with external environments) are misaligned, so joint multi-task training yields unstable behavior and limited gains; M2A instead works in parameter space, identifying the feature subspace critical for agent behavior and merging the mathematical reasoning task vector only along its null space so reasoning is injected in directions that do not perturb agent behavior.
- The merge needs no additional gradient updates and exposes the merging coefficient as a knob for reasoning length; applied to a fine-tuned Qwen3-8B coding agent, it raises SWE-Bench Verified resolved rate from 44.0% to 51.2% without retraining the model.
- As coding agents grow more tool-driven, the question shifts from whether a model can reason harder to whether new reasoning skills can be added without breaking the interaction patterns that make agents useful.

Application-Layer Innovations

PAPERS 7 - 12

Chroma Context-1: Training a Self-Editing Search Agent

trychroma.com/research/context-1 ↗

Context-1 is a 20B agentic search model derived from gpt-oss-20B that is designed to act as a retrieval subagent rather than answer the question directly.

- It decomposes a task into subqueries, searches over multiple turns, and selectively prunes its own context to remove irrelevant documents.
- Trained on more than 8,000 synthetic tasks with a curriculum that shifts from recall-heavy exploration toward higher-precision retention, it sits on the report's cost-latency frontier and improves prune accuracy from 0.824 to 0.941.
- Multi-hop retrieval may work best as a specialized agent problem, where small purpose-trained models learn to plan, explore, and actively edit context instead of relying on single-pass RAG pipelines.

Composer 2 Technical Report

arxiv.org/abs/2603.24477

Composer 2 is a domain-specialized model for agentic software engineering, built to handle long-horizon coding tasks while staying efficient enough for interactive use.

- Its training recipe combines continued pretraining on a code-heavy mix and a 256k context extension with targeted coding SFT, then follows with large-scale reinforcement learning in Cursor's production tool-use harness.
- The report introduces CursorBench, an internal benchmark drawn from real engineering sessions rather than curated public repos; Composer 2 scores 61.3 on CursorBench-3 and is positioned on a strong accuracy-cost frontier relative to larger general-purpose models.
- Coding agents improve most when the model, training environment, reward design, and evaluation all match the actual software-engineering workflow.

GEPA: Reflective Prompt Evolution Can Outperform RL

arxiv.org/pdf/2507.19457

GEPA (Genetic-Pareto) is a reflective prompt-evolution method that optimizes instructions using execution traces instead of updating model weights.

- The system samples rollouts, has the model analyze its own traces in natural language, and proposes new prompts; a Pareto front keeps multiple candidates that perform well on different subsets of data.
- Across four tasks, GEPA matches or beats GRPO-style RL with up to 35× fewer rollouts.
- The work treats prompts as an external optimization layer, using natural-language reflection rather than heavyweight RL.

Dockerless: Environment-Free Program Verifier for Coding Agents

arxiv.org/abs/2606.28436

Dockerless is an agentic patch verifier that evaluates coding-agent outputs without running repository-specific test environments.

- Instead of executing unit tests inside per-repository Docker images, it judges patch correctness from evidence gathered through agentic repository exploration, rather than matching candidate patches against references.
- On a verifier evaluation benchmark it outperforms the strongest open-source verifier by 14.3 AUC points, and when used as both the SFT trajectory filter and RL reward in a fully environment-free post-training pipeline the resulting model reaches 62.0% on SWE-bench Verified, 50.0% on Multilingual, and 35.2% on Pro.
- As coding-agent training becomes bottlenecked by verification cost, agents that can reliably judge patches without rebuilding every target environment make post-training loops faster, cheaper, and easier to scale.

MEM1: Constant-Memory Long-Horizon Agents

arxiv.org/html/2506.15841v1 ↗

MEM1 is an RL framework that trains LLM agents to operate over long multi-turn tasks while keeping memory usage nearly constant.

- At each step, previous memory and new observations are merged into a compact internal state token (IS) and older context is discarded; a masked-trajectory RL scheme reconstructs valid trajectories for PPO without feeding the entire history.
- MEM1-7B matches or beats much larger baselines on tasks with up to 16 sequential objectives while reducing memory use by ~3.7×.
- Long-horizon behavior can come from learned internal state handling rather than expanding context windows or bolting on external memory.

SWE-Explore: Benchmarking How Coding Agents Explore Repositories

arxiv.org/abs/2606.07297 ↗

SWE-Explore is a benchmark that isolates the repository-exploration step of coding agents by scoring how well they find the relevant code before any fix is attempted.

- Given a repository and an issue, it asks an explorer to return a ranked list of relevant code regions under a fixed line budget, and scores that output along coverage, ranking, and context-efficiency dimensions.
- The benchmark spans 848 issues across 10 programming languages and 203 open-source repositories, with line-level ground truth derived from independent agent trajectories that solved the same issue; agentic explorers form a clear tier above classical retrieval, and while file-level localization is already strong, line-level coverage and efficient ranking remain the axes that differentiate state-of-the-art explorers.
- Repository-scale coding agents are limited not only by generation quality but by how well they spend a fixed context budget on the specific files and lines a correct patch depends on.

ABOUT THIS REPORT

Built from real engineering data

The State of AI Coding is a quarterly Greptile Research publication tracking how AI is changing software development. Figures are drawn from Greptile's merge-request dataset and from public package-registry download data, using the methodology described on page 02.

Greptile reviews pull requests with full-codebase context, catching bugs and enforcing team standards automatically. The engineering-velocity trends in this report are measured on the same infrastructure.

[READ ONLINE](#)www.greptile.com/state-of-ai-coding

© 2026 GREPTILE · PUBLISHED JULY 2026