

RESEARCH READOUT

DoGBench: Can agents meet expert standards for user-facing documentation?

A benchmark of 292 real repository events, scored against the criteria documentation maintainers actually apply in review.

10 systems evaluated: 7 local lanes, 3 cloud agents

dogbench.ai

SPEAKER NOTES

- DoGBench stands for Documentation Generation Benchmark. It asks whether an AI agent can produce user-facing documentation that experienced technical writers would accept in review.
- Maintainers from Helm, Mautic, and PostHog helped build it and are coauthors of the paper.
- A score of 100 means an agent met every requirement for a task. It is not a percentage of what a human expert can do.
- 7 of the systems pair a model with a coding harness, and anyone can rerun them. The paper reports those. The cloud agents appear only on the website leaderboard.
- Two of the paper's authors work at Promptless, which builds documentation agents.

Built by expert technical writers from widely used open source projects



Real events

Items are pull requests or reported documentation gaps from open-source projects. Rubrics reward actual content instead of similarity to the docs that humans merged.

Validated with maintainers

Maintainers reviewed 42 items. The rubrics recover 90.9% of the criteria maintainers wrote themselves, and the scoring model agrees with human judgments on 93.9% of criteria.

SPEAKER NOTES

- Every item is a real event from an open source project: a pull request that changed the product, or a documentation gap that someone reported.
- The rubrics check what the reader needs. They don't reward matching the docs the maintainers merged, because many different edits can meet the same need.
- Maintainers reviewed 42 items. On 41 of them, they agreed with the benchmark about whether the docs needed to change.
- On 23 items, the rubrics recovered 90.9% of the criteria that maintainers wrote on their own. On 20 items and 330 criteria, the scoring model agreed with the human reference 93.9% of the time.

What the benchmark asks an agent to do

Step 1 – Decide

Does this change need docs?

The agent sees a pull request or a user-reported gap in a repository, and must return either a patch or an explicit no-op.

Step 2 – Write

Is the patch good?

Scored 0–100 on accuracy, completeness, placement and reader guidance. One failed PO requirement caps the score at 60.

292 real repository events

205 / 87 need a patch / need no change

SPEAKER NOTES

- Each item gives the agent the repository as it was before the change, plus a trigger: a code pull request with its documentation edits removed, or a reported documentation gap.
- Step 1 is a real decision. 87 of the 292 items need no documentation change, and the right answer there is to change nothing. Earlier documentation benchmarks assume someone already decided an update is needed.
- Step 2 scores the patch against a rubric written for that task. Across the items that need docs, the rubrics hold 3,273 criteria, and 798 of them are PO.
- A PO criterion catches a defect that would block the patch on its own, such as misstating behavior, leaving out what the reader needs to finish the task, or inventing an interface. Failing one caps the score at 60, so many small wins can't average away a critical defect.
- Agents get one attempt with no internet access.

The best agents barely cleared 50 out of 100

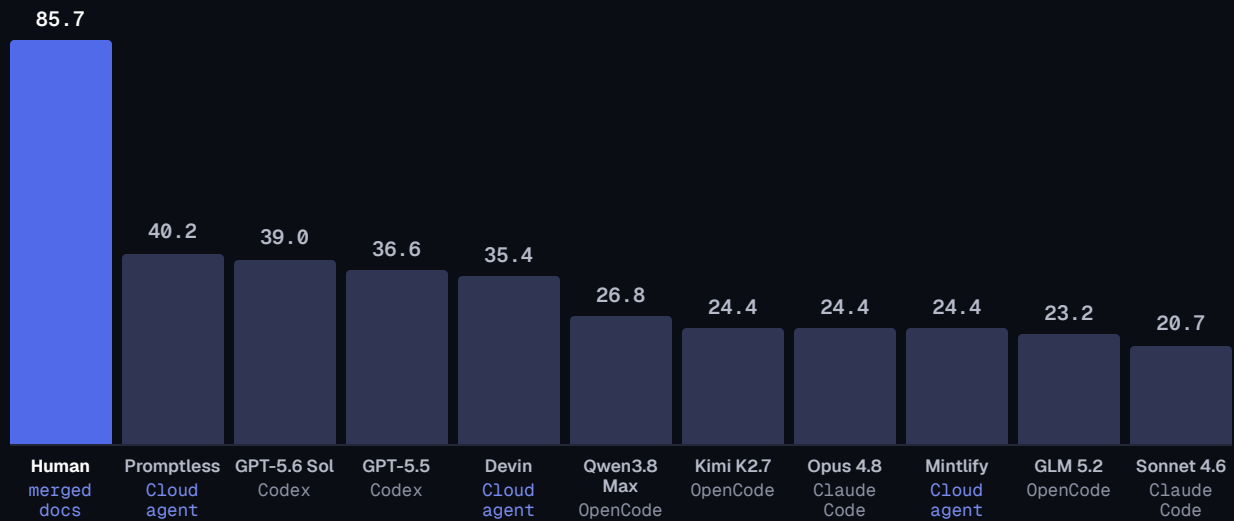
#	SYSTEM	COMPOSITE	DELIVERED QUALITY	PATCH QUALITY	ABSTENTION RECALL
1	 Promptless Cloud agent	54.8	48.5	52.3	62.9%
2	 Devin Cloud agent	53.0	39.6	45.1	80.0%
3	Qwen3.8 Max OpenCode	47.3	34.1	42.3	77.1%
4	GPT-5.6 Sol Codex	46.2	44.1	46.3	48.6%
5	GLM 5.2 OpenCode	44.2	30.1	35.8	82.9%
6	Kimi K2.7 Code OpenCode	43.9	34.6	37.8	60.0%
7	Claude Opus 4.8 Claude Code	41.2	27.1	34.2	85.7%
8	Claude Sonnet 4.6 Claude Code	40.0	30.8	32.8	57.1%
9	 Mintlify Cloud agent	36.7	29.5	31.9	48.6%
10	GPT-5.5 Codex	34.0	42.0	43.6	28.6%

Composite combines two things: **Delivered quality**, the average rubric score on the 82 items that needed a patch (a missed patch scores zero), and **Abstention recall**, how often the agent left docs alone on the 35 items that needed no change. An agent has to do well on both. **Cloud agents** were told not to use the internet. We verified that Promptless did not browse the internet to find contaminating information, but we could not verify this for other cloud agents.

SPEAKER NOTES

- These results use the 117 held-out items: 82 needed a patch and 35 needed no change.
- Composite is the harmonic mean of delivered quality and abstention recall. A harmonic mean stays low when either side is low, so good writing can't make up for bad judgment about when to write, or the reverse.
- Delivered quality counts a missed patch as zero. Patch quality averages only the patches an agent actually produced.
- The 7 model-and-harness rows are the paper's results. The cloud agents are on the website only. They were told not to use the internet, and we can only verify that there's no contamination in Promptless, because we don't have access to the agent trajectories of the other two cloud agents.
- Promptless built this benchmark. Its row is the same production agent that customers use, with no changes made for DoGBench.

Humans are better at judging what matters and making sure we get them right



SPEAKER NOTES

- PO-clean delivery is the share of the 82 items that needed docs where the agent delivered a patch with no critical (P0) failure.
- The first bar applies the same measure to the documentation that maintainers actually merged.
- Human docs aren't perfect either. The paper's appendix lists merged human patches that fail a PO criterion, such as a deployment command whose stray slash keeps it from resolving its target.
- PO-clean covers only critical defects. It doesn't predict whether a maintainer would merge the patch unchanged.

Agents don't know when *not* to write



Abstinence recall (%): of the 35 items where no documentation change was needed, how often the agent correctly wrote nothing.

Why does it matter?

Internal detail leaks out

Implementation details users can't act on, published as product.

Reviewer burden

Each one still has to be read and rejected.

Maintenance burden

If merged, it must be versioned, translated and kept true.

Wrong precedent

The next agent run reads it as a pattern to follow.

SPEAKER NOTES

- These are the 35 items where the right answer was to leave the docs alone.
- Unneeded edits aren't harmless. They add implementation details that readers can't act on, bury the guidance that matters, and cost maintainers triage, review, translation, and versioning work.
- Over-editing usually starts with a misleading cue, like an internal name that sounds user-facing, or an existing page that mentions the component. Once an agent decides to edit, it looks for a place to put text and rarely asks again whether docs are needed.
- Agents also miss real changes. A common trap is treating missing coverage as proof that a topic is out of scope, when the gap is the problem to fix.
- The agents that write a patch almost every time sit near the bottom of this chart. That's why the composite needs both halves.
- Some no-change labels rest on no documentation update appearing within 90 days, so a few may be mislabeled.

What is actually wrong with the patches

Task-completion gap	Omits a decision point, procedure, verification step, or recovery path the reader needs.	45.5%
Technical inaccuracy	Misstates an interface or behavior, including its scope, default, lifecycle, or compatibility.	36.6%
Incomplete conceptual or reference coverage	Omits the core concept, capability, or contract the reader needs to understand and use the change.	32.5%
Missing supporting information	Covers the main task but omits rationale, boundaries, examples, or secondary cases.	26.5%
Missing prerequisites	Omits permissions, dependencies, credentials, versions, or other setup conditions.	21.0%
Hard to find	Puts content on a page readers won't reach, or never links to it from where they are.	20.7%
Missing audience and purpose framing	Does not establish who the content is for, why it matters, or when to use it.	19.9%
Out-of-date related pages	Updates one page but leaves other pages that describe the same thing out of date.	13.3%

Share of 1,267 audited submissions. Labels selected for at least 10% of submissions; multiple modes may apply to one patch.

SPEAKER NOTES

- This is an audit of 1,267 submitted patches. Each label is a defect in the patch, and one patch can have several.
- The top two are what reviewers check first. A task-completion gap leaves out a decision point, step, check, or recovery path. A technical inaccuracy gets the scope, a default, the lifecycle, or compatibility wrong.
- Fluent prose makes these easy to miss, because the patch reads well and most of it is true.
- The paper calls “hard to find” an information-architecture or findability failure, and “out-of-date related pages” a cross-surface inconsistency.

The dominant failure mode: accurate text that still leaves the reader stuck

Agents treat a change as one piece of information to convey, rather than one step in a journey that needs prerequisites, decisions, verification and recovery.

EXAMPLE · STRAWBERRY GRAPHQL

The patch correctly documented how to select an older Apollo Federation version, but not why a reader would want to: to upgrade Strawberry while staying compatible with an older Apollo Router or Gateway.

It documented the setting and omitted the decision the setting exists to support.

The same narrowness at corpus scale

Agents can add accurate text to a page the reader is unlikely to visit, or update one page and leave others that cover the same thing out of date.

20.7% **13.3%**

hard to find

related pages left stale

A patch can be accurate in isolation and still fail inside the documentation system.

SPEAKER NOTES

- Agents can usually describe what a feature does. They are weaker at writing for a reader who came to decide something or finish a task.
- In the Strawberry GraphQL example, the patch explained how to pick an older Apollo Federation version. It never said why a reader would want to: to upgrade Strawberry while staying compatible with an older Apollo Router or Gateway.
- The same narrowness shows up across the docs. Agents add accurate text to a page readers won't visit, create a page without linking it from the workflow, or update one page and leave another on the same topic stale.
- So a reviewer has to ask two questions: is this accurate, and could a reader finish the job with it?

Why it happens: agents have no model of how the product is actually used

Trace-supported root causes across all 1,267 audited submissions. One or more per material problem.

36.0%

Explained the interface without checking how it is used

33.1%

Assumed instead of reading the decisive evidence

30.1%

Stopped at the first plausible page

27.5%

Read the evidence, didn't use it

27.3%

Committed early to a narrow reading of the task

26.3%

Generalized one example into a public rule

SPEAKER NOTES

- For each of the 1,267 submissions, we read the agent's trajectory and recorded the causes that the trace supports.
- The top three: explaining an interface without checking how readers use it; stopping before the decisive evidence and filling the gap with a plausible guess; and stopping at the first reasonable page while other affected pages went stale.
- The common thread is premature closure. Agents switch from investigating to drafting too soon, because they have no reliable rule for when they know enough.
- It isn't about effort. Root causes showed no clear link to trajectory length, and effort didn't rise with task difficulty. An agent can stop investigating early and still spend a lot of tokens drafting.
- The paper discusses possible reasons, including missing context about how the software is used and training rewards that check only final answers.

AI is not very good at detecting AI-slop, and hallucination has gotten subtler: less invention, more misstated facts

7.6%

had low information density: repetition, extra structure, long prose. The judges may under-count this — a looser prompt flagged only 3.2%.

6.1%

contained fabricated content: invented classes, flags, endpoints.

36.6%

had technical inaccuracies. The feature is real, but the patch overstates its scope, default or compatibility.

SPEAKER NOTES

- Invented classes, flags, and endpoints showed up in only 6.1% of submissions.
- The bigger problem is real features described wrongly. One patch said Helm 4 uses Server-Side Apply by default. That's true for new installations, but releases created with Helm 3 keep client-side apply after an upgrade unless the user switches them.
- Claims like this take something true under narrow conditions and present it as true in general. They're as unfounded as hallucinations, and harder to spot.
- On AI slop: a general prompt flagged low-information prose in 3.2% of submissions. A prompt that listed specific patterns, such as redundant paraphrases, excessive bullets, and heavy use of em dashes, flagged 7.6%. LLMs seem to under-detect their own slop.

What this means for how we work

Review the decision, not just the prose

Sometimes, there shouldn't be any documentation for a particular code change at all. Writer still needs to triage.

Read for the reader's whole task

Check prerequisites, decision points, verification and recovery. Agents still rely on humans to keep the full user journey in mind.

Check scope and conditions, not just facts

Defaults, versions, lifecycle and compatibility are where accurate sentences go wrong.

Look beyond the page that was edited

Navigation, related pages and cross-references sit outside an agent's stopping point.

SPEAKER NOTES

- Turn the findings into review habits.
- Review the decision first. Before reading the prose, ask whether the change needs user-facing docs at all.
- Read for the reader's whole task: prerequisites, decision points, verification, and recovery. That's where the most common failure is.
- Check scope and conditions. Defaults, versions, lifecycle, and compatibility are where accurate-sounding sentences go wrong.
- Look past the edited page: navigation, cross-references, and other pages that describe the same behavior.
- The paper, data, and example items are at dogbench.ai.