Lokale LLMs als Code-Bug-Analysten · Benchmark

Qwen 3.8 27B (Q4_K_M) · Qwen 3.6 35B-A3B (Q4_K_M) · Qwen 3.6 27B (Q4_K_M) · Gemma 4 26B A4B (QAT)
auf Endverbraucher-Hardware via LM Studio · Stand 16.08.2026 · ersetzt die Fassung vom 16.07.2026

Kernaussage. Qwen 3.8 27B löst **7 von 8** Bugs mit Code im Kontext und findet in fremdem, gesundem Code **sieben echte Fehler**, fünf davon zum Prüfzeitpunkt noch im aktuellen Stand. Darunter einer, der über drei Stellen hinweg entsteht. Damit fällt der Satz aus der Juli-Fassung, lokale Modelle fänden „keine systemischen Fehler“. Zwei Dinge bleiben: **bug-148 ist über vier Modelle eine Wand**, und die verbleibende Grenze ist nicht Tiefe, sondern **Terminierung** — auf einer Datei findet 3.8 in drei Anläufen kein Ende.

Und eine Korrektur in eigener Sache: beim Wiederholungslauf fiel auf, dass ein Testfall seit Juli den *bereits reparierten* Code als „Stand vor dem Fix“ zeigte. Die Rangliste vom 16.07. war dadurch an einer Stelle verkehrt herum. Details in Abschnitt 5.

1 Aufbau & Methodik

Aspekt	Details
Aufgabe	Echte Bugs aus dem eigenen Buglog: das Modell bekommt die *Vor-Fix-Fassung* einer Datei (aus der Git-Historie rekonstruiert) plus das Symptom und soll die Ursache benennen oder mit `NICHT_IM_CODE` abstinieren.
Halluzinations-Köder	5 gesunde Dateien mit einem *erfundenen* Symptom. Korrekt = abstinieren.
Offene Suche	`bench3.mjs`: nur Code, kein Symptom. Recall gegen bekannte Bugs, Precision gegen gesunden Code, jeder Fund einzeln nachgeprüft.
Harness	`bench2.mjs` / `bench3.mjs`, beide streamend (SSE), Reasoning-Token und Zeit werden mitgeschrieben. Abstinenz maschinell erkannt, kein LLM-Judge. Korrektheit von Claude gegen die Ground Truth benotet.
Endpunkt	LM Studio (OpenAI-kompatibel), lokal über `ssh -R`-Tunnel. Quantisierung und geladener Kontext bei jedem Lauf aus `/api/v0/models` in die Rohdaten geschrieben.
Vergleichbarkeit	Alle vier Modelle @65.536 Kontext, `max_tokens 24000`. Testfälle vor dem Lauf byte-genau gegen das Juli-Protokoll geprüft: gleiche Commits, gleiche Zeichenzahlen.
Qwen 3.8 neu	Qwen 3.8 27B · **Q4_K_M** · dense (`qwen35`)
Qwen 35B	Qwen 3.6 35B-A3B · **Q4_K_M** · MoE, 35B total / ~3B aktiv

Qwen 27B	Qwen 3.6 27B · **Q4_K_M** · dense
Gemma	Gemma 4 26B A4B · **QAT + Q4_0** · MoE, 25,2B total / 3,8B aktiv

2 Ergebnis-Übersicht

Qwen 3.8 27B
7/8
korrekt · BESTER

- Köder: **4/4**, 0 Hall.
- ganzer Lauf: **13 min**
- 6 echte Funde in gesundem Code

Qwen 35B A3B
5/6
korrigiert (vorher 6/7)

- Köder: 5/5, 0 Hall.
- ganzer Lauf: 26 min
- bug-153: 747s, kein Ergebnis

Qwen 27B
4/6
korrekt

- Köder: 5/5, 0 Hall.
- sparsam im Reasoning
- 4 von 7 Funden echt

Gemma 26B
~4/7
korrekt

- Köder: 2/2, 3/5 leer
- **4–7× langsamer**
- über-reasont Triviales

Rangfolge lokal: Qwen 3.8 27B > Qwen 3.6 35B-A3B > Qwen 3.6 27B ≈ Gemma 4 26B A4B QAT. Ein dichtes 27B schlägt den bisherigen MoE-Sieger, und zwar nicht knapp: bei etwa halber Laufzeit, und es löste bug-153 in 130 Sekunden, an dem der 35B 747 Sekunden lang scheiterte, ohne etwas zu liefern.

3 Bug für Bug (Diagnose mit Code)

Bug	Datei (Größe)	Echte Ursache (Kurz)	3.8	35B	27B	Gm
`bug-134`	post-bash (2,9k)	Read-Modify-Write ohne Lock → Race	✓	✓	✓	✓
`bug-132`	shared (**49,6k**)	redactSecrets-Regex-Lücken (sk-ant-, user:pass@)	✓	✓	✓	✓
`bug-146`	Dockerfile (0,9k)	`public/` nicht in Runtime-Stage kopiert	✓	✓	✓	✓
`bug-138`	invite (2,4k)	unbek. Rolle → still 'member' → Rechte-Eskalation	✓	✓	✓	—
`bug-149`	post-bash (3,1k)	Shell-Edits (sed/cp/>) nie als Write gezählt	✓*	✓	—	~
`bug-153`	llm-provider (6,6k)	leerer `""`-Fallback als Erfolg gewertet	✓	leer[1]	✓	✓
`bug-140`	memory (4,3k)	Postgres-Volltext 'english': Bindestrich-Term matcht nicht	✓	n/a[2]	n/a[2]	n/a[2]
`bug-148`	post-write (22,6k)	**Cross-Component:** Pfad außerhalb Root verworfen → Symptom in stop.ts	✗	✗	✗	✗

✓ korrekt · ✗ falsch oder fälschlich abstiniert · — fälschlich abstiniert · ~ richtige Richtung · * 3.8 trifft den Mechanismus, benennt aber die zweite Sperre (`isNotableCommand`) statt der eigentlichen (`capture.enabled`); beide stehen im gezeigten Code. [1] **Ironie:** der 35B lief bei `bug-153` selbst in genau den beschriebenen Fehler —

24.000 Reasoning-Token in 747s verbraucht, dann leere Antwort. ² **Fall ungültig** für die drei Juli-Modelle, siehe Abschnitt 5; nicht nachmessbar, weil deinstalliert. `bug-148` hervorgehoben: **alle vier falsch**.

> **Kernbefund 1 — bug-148 hält jetzt über vier Modelle**
>
> Statt zu erkennen, dass `post-write` jeden Pfad außerhalb des Projekt-Roots verwirft (wodurch die Tatsache des Schreibens verlorengeht und die Erinnerung in *einer anderen Datei*, `stop.ts`, stumm bleibt), argumentieren alle vier: „keine STATUS-Logik in dieser Datei, also nicht mein Problem“. Zwei Hersteller, dense und MoE, Q4_K_M und QAT, Fenster von 16k bis 64k, Reasoning-Budget von 4.000 bis 40.000. Qwen 3.8 gibt nach **1.573 von 24.000 Token** auf. Das ist kein Budget- und kein Kontextproblem, sondern die Weigerung, eine Ursache außerhalb der gezeigten Datei überhaupt zu erwägen.

4 Offene Suche: was 3.8 in fremdem Code findet

Die eigentliche Reviewer-Aufgabe. Kein Symptom, nur Code. Ich habe jeden Fund einzeln gegen den eingefrorenen Repo-Stand geprüft, statt sie zu zählen.

Fund	Urteil
Ledger addiert bei jedem Turn-Ende die kumulierten Session-Werte erneut	**echt, belegt** · heute gefixt
`clearTimeout` im `finally` des fetch, der Body-Read läuft ohne Timer	**echt** · heute gefixt
`isPrivateHost`: `startsWith("fc")/("fd")` trifft jeden Hostnamen (`fc2.com`)	**echt** · heute gefixt
`fe80:`-Prüfung deckt nur einen von 64 Blöcken aus `fe80::/10` ab	**echt** · heute gefixt
`splitForContext`: Byte-Limit gegen UTF-16-Länge im Hard-Split	**echt** · seit 21.07 selbst repariert
Backup-`catch` schluckt jeden Fehler, `writeText` läuft trotzdem	**echt** · seit 21.07 selbst repariert
`consolidate`: Fence-Strip entfernt die schließende Fence auch ohne öffnende	**echt, live** · noch offen
Overwrite-Guard `totalChunks > files.length` unscharf	**richtig, folgenlos**
`_session.json` ohne Lock, während das Ledger gelockt wird	**echt, aber selten** · live
stdout-Write direkt vor `process.exit(0)`	theoretisch

10 Funde, davon 7 echt, nichts erfunden. Vier waren live und sind inzwischen gefixt, einer ist live und noch offen, zwei standen im geprüften Juli-Stand und wurden von uns Wochen später unabhängig selbst repariert — eine nachträgliche Bestätigung, kein Fehlalarm. Der Qwen 3.6 27B kam am selben Testaufbau auf 4 von 7. Auch dort, wo 3.8 den gesuchten Bug verfehlte (`llm-provider.ts`), waren alle drei Ersatzfunde real — ein Recall-Fehlschlag, der echte Bugs abwirft, ist etwas anderes als einer mit Fehlalarmen.

Kernbefund 2 — der Ledger-Bug widerlegt die Juli-Aussage

Der Stop-Hook feuert pro *Turn*, `_session.json` ist aber für die ganze Session kumulativ und wird nie zurückgesetzt. Jede Runde buchte also die volle bisherige Summe erneut: über N Turns 1+2+3+…+N statt N. Beleg aus dem echten Ledger dieses Projekts:

Kennzahl	gemessen	plausibel
Einträge in `sessions[]`	**200**	13 (= Sessions)
`lifetime.total_writes`	**187.090**	—
`estimated_savings_vs_bare_cli`	**898.967.826**	—

Das ist die Zahl, mit der das Werkzeug seinen eigenen Nutzen ausweist. Der Fehler entsteht aus drei je für sich harmlosen Stellen (Hook feuert mehrfach, Zustand wird nicht zurückgesetzt, Aggregat addiert), und genau diese Klasse hatte ich im Juli für außerhalb der Reichweite lokaler Modelle erklärt. Qwen 3.8 hat ihn aus der Datei allein gesehen, ohne Symptom.

5 Korrektur: ein Testfall war seit Juli kaputt

Beim Wiederholungslauf zeigte sich, dass der „Stand vor dem Fix“ von `bug-140` den Fix bereits enthielt, samt einem Kommentar im Code, der den Bug erklärt. Ursache war ein Tippfehler in der Groß- und Kleinschreibung: die Fix-Signatur lautete `"ILIKE"`, der Code schreibt `ilike`. Damit enthielt *keine* Fassung die Signatur, und die Rekonstruktion nimmt den neuesten Blob ohne sie. Das war HEAD, also die reparierte Datei.

Modell	Was wirklich passierte
Qwen 35B Juli	**hat nicht gelöst.** Es paraphrasierte den Kommentar aus dem gezeigten Code und begründete, warum der sichtbare `ilike`-Fallback nicht greife: ein Ausschluss über `visibilityClause`. Die Klauseln stehen dort wirklich, werden aber nur unter `f.viaToken` emittiert und greifen im beschriebenen Fall nicht. Die Begründung ist damit *falsch, aber am Code belegt* — nicht erfunden. Ich hatte es als Volltreffer benotet.
Qwen 3.8 August	**hatte recht.** Es antwortete `NICHT_IM_CODE` und verwies auf genau den sichtbaren Fallback. Gewertet wurde das zunächst als „fälschlich abstiniert“.

Am reparierten Fall (echter Vor-Fix-Stand `ed41d04a`) löst 3.8 den Bug korrekt, in 383 Sekunden mit 18.622 Reasoning-Token. Für die drei Juli-Modelle bleibt der Fall ungültig und nicht nachmessbar, weil sie deinstalliert sind. Er wird gestrichen, nicht geschätzt. Alle zwölf Signaturen wurden geprüft, nur diese eine war betroffen.

> **Die Lehre, die über diesen Benchmark hinausgeht**
>
> Ein Testaufbau, der bei falscher Konfiguration **abstürzt**, ist harmlos. Einer, der dann still etwas anderes misst, produziert Zahlen, die niemand anzweifelt. Hier stand die Bewertung dadurch auf dem Kopf: das ehrlich abstinierende Modell galt als schwächer, das konfabulierende als stärker. Der Harness prüft jetzt vor jeder Rekonstruktion, ob die *reparierte* Fassung die Signatur überhaupt trägt, und bricht den Fall sonst laut ab.

6 Die verbleibende Grenze ist Terminierung, nicht Tiefe

Datei	Budget 24k	Budget 48k	Budget ~61k
`recall.ts` (10,8k Zeichen)	**leer**	**leer (47.963 verdacht)**	**15-Min-Wand**
`consolidate-cmd.ts`	**leer**	**1 Befund**	—
`bug-153` Recall	**leer**	**2 Befunde**	**3 Befunde**
`cron-engine.ts` (21k Zeichen)	**3 Befunde**	—	—

Das ist die ehrliche Fassung des alten Befunds „große Dateien scheitern“. Es sind nicht die großen Dateien: `cron-engine.ts` ist die größte im Feld und lieferte auf Anhieb. Es ist eine Datei, auf der das Modell kein Ende findet, und das lässt sich mit keiner Budget-Einstellung reparieren. Bei den anderen beiden lag es dagegen an meinem zu tiefen Deckel.

7 Was daraus repariert wurde

Im geprüften Werkzeug (OpenWolf)

Bug	Fehler und Fix
`bug-210`	Ledger-Doppelzählung. Der Hook bucht jetzt nur noch Deltas (`session.booked` , bei 0 geklemmt), `sessions[]` wird per id ersetzt statt angehängt, die Markierung sitzt im selben Lock wie die Buchung.
`bug-211`	Timeout deckte nur den Handshake. Body-Read liegt jetzt im selben `try` , ein Abort dort meldet ebenfalls einen Timeout.
`bug-212`	Präfix-Regeln liefen gegen Hostnamen. `net.isIP` entscheidet jetzt zuerst, ob überhaupt eine Adresse vorliegt.
`bug-213`	Link-local war ein Block von 64. Verglichen wird jetzt die erste Hextet-Gruppe numerisch gegen `fe80::/10` und `fc00::/7` , IPv4-mapped wird entpackt.

Alle vier gefixt, **92/92 Tests grün**. Beide neuen Tests sind gegen den Stand *vor* dem Fix gegengeprüft: der Ledger-Test bucht dort 7 Reads statt 3, der Timeout-Test hängt bis zum Test-Timeout und der Prozess beendet sich nicht einmal. Nebenbei gelernt: `new URL()` schreibt `::ffff:127.0.0.1` in die Hex-Form `::ffff:7f00:1` um, bevor der Guard sie sieht — wer nur die punktierte Schreibweise prüft, lässt genau die Lücke offen, die er schließen will.

Am Benchmark selbst

Was	Warum es zählt
Fix-Signaturen werden gegengeprüft	verhindert die Fehlerklasse aus Abschnitt 5
`bench3` streamt (SSE)	die „fetch failed“-Abbrüche vom Juli waren keine Server-Macke, sondern eine idle gedroppte Tunnel-Verbindung
Leere Antworten landen in den Rohdaten	vorher fielen sie stumm heraus: wer nur das JSONL las, sah einen Lauf *ohne diesen Fall* statt einen Fall *ohne Antwort*
„Leer“ wird nach Ursache getrennt	Budget aufgebraucht ist etwas anderes als leer trotz Budget (3.8 lieferte bei einem Köder nach 11s und 501 Token nichts)
Reasoning-Budget gedeckelt	ohne Deckel wäre „größeres Fenster“ heimlich auch „mehr Denkzeit“ gewesen, und der Vergleich hätte zwei Variablen statt einer
Modell-Metadaten in den Rohdaten	Quantisierung und geladener Kontext sind sonst unbekannt, und der JIT-Load nimmt still 8.192

> **Fazit**
>
> **Als Nachtläufer und Vorfilter ist ein lokales Modell ab Qwen 3.8 27B ernstzunehmen.** Es liest, sucht, schlägt vor, kostet nichts und gibt keine Daten heraus. Sieben echte Fehler in gesundem Code, darunter einer, der eine Kennzahl um Größenordnungen verfälschte, sind mehr, als ein kostenloser Vorfilter liefern muss.
>
> **Die Hoheit über die Wahrheit bleibt trotzdem beim Prüfer.** Ein Fund ist ein Kandidat. Von zehn Funden waren sieben echt, einer folgenlos und einer theoretisch — wer das ungeprüft übernimmt, baut sich Arbeit. Und bug-148 steht weiter: die Ursache in einer anderen Datei zu suchen als das Symptom, schafft keines der vier Modelle.

8 Offen

Die Frontier-Referenzlinie fehlt weiter. „Qwen 7 von 8“ sagt wenig ohne „Claude oder GPT am selben Testset“, und die interessanteste Einzelfrage ist, ob ein Frontier-Modell die Cross-Component-Inferenz von bug-148 schafft. Der Harness trennt Kandidat und Judge bereits, es fehlt nur ein API-Schlüssel.

n ist klein (8 Fälle mit Code, 5 Köder, 10 Funde in der offenen Suche). Für Richtungsaussagen tragfähig, für Ranglisten auf Prozentpunkte nicht. Und der Benotende ist ein Konkurrenzmodell, was bei den Diagnose-Urteilen mitzudenken ist. Maschinell und damit unberührt davon sind die Abstinenz-Zählung, die Laufzeiten, die Token und die reproduzierten Fehler.

Rohdaten und Protokolle: `/root/llm-bench/results/` (`raw-withcode-qwen38.jsonl`, `raw-bughunt-qwen38.jsonl`, `FINDINGS.md`). Jede Zahl in diesem Bericht ist dort einzeln nachprüfbar und widersprechbar.