Glowing Evals

27 May 2026 · ai, testing, evals, banking, critical thinking

There's a fascinating scene in the series "Chernobyl", where the engineers report that the radiation level is 3.6 roentgen (https://en.wikipedia.org/wiki/Roentgen_(unit)). The number, they explain, is high but not terrible for a nuclear accident. Initially it's reported up to soviet premier Gorbachev as the equivalent of a chest X ray, soon to be corrected as actually 400 chest X rays, far from ideal. But the protagonist, Professor Legasov, realises this is also probably wrong, but for another reason: 3.6 was the maximum level that low level dosimeters could detect.

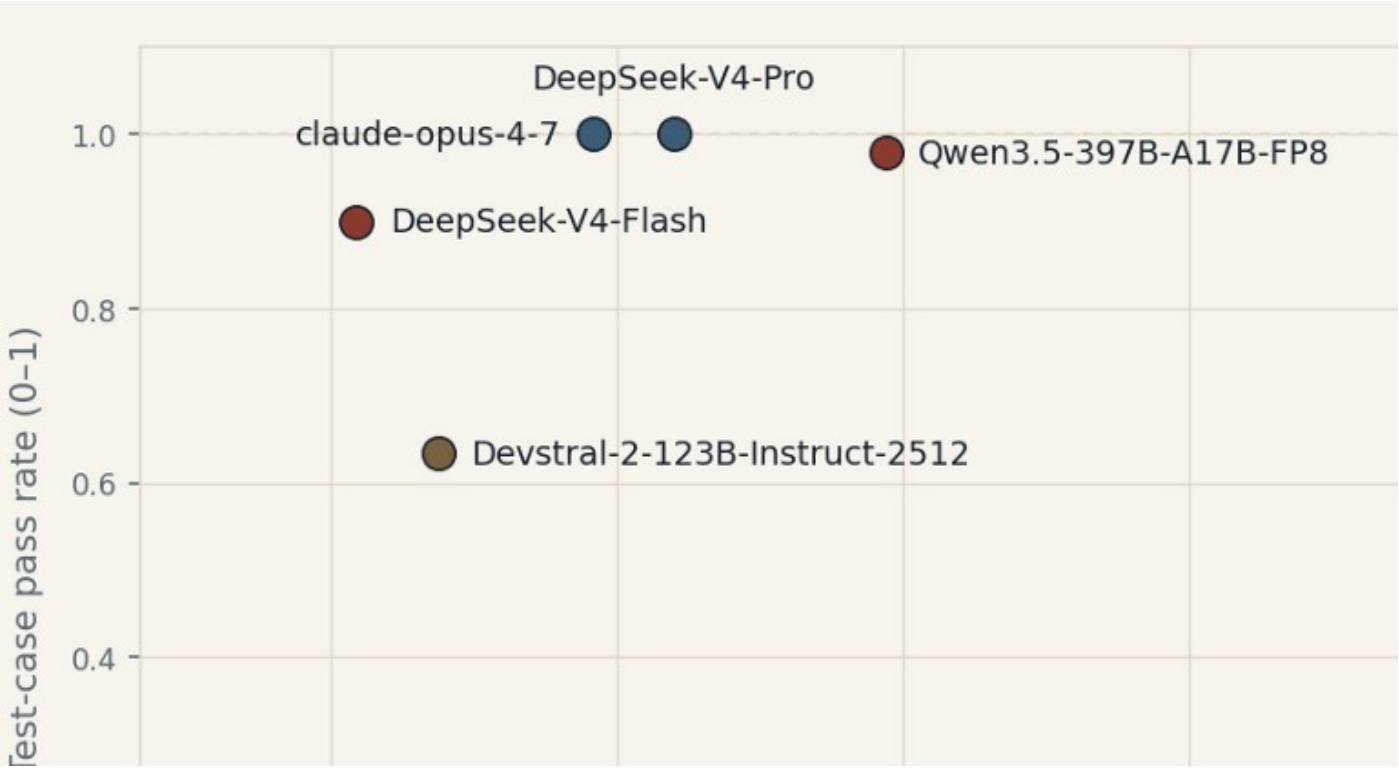
Being correctly able to assess what your software & agents can do is a lifesaver

The true value might be 3.6 roentgen or even higher, but the detector couldn't report a higher number even if that was the case. Subsequent analysis of the Chernobyl site showed an actual value of 15000 roentgen. This ties in with a common bias in how engineers often approach software testing. The bias is to look for simple cause or explanation. "It's just a flaky

test” or “something’s timed out”, “Its a one off” etc. This leads to issues not being investigated further.

But that whole approach is deeply flawed. why are we assuming that the error we are seeing I not the tip of the iceberg? from our limited vantage point we might just see a “minor” ‘issue. To paraphrase a scene from father Ted: its big, but it’s just very far away.

But even if we now look at AI evals, sometimes used to show “how well” a model or model + context is doing at a task, we see a similar issue. Though one that is often invisible without the relevant training and/or experience. Below is the sort of chart I create for some evals:



We can’t tell if Deepseek or Claude Opus is better, they both score ≈1/1

If my eval’s reported score, possibly calculated from many individual executions is 100%, for configuration X but 80% for configuration Y. then many tend assume we have correctly identified the capability of our model/intent/agent, Though just as our Chernobyl dosimeter can’t measure the true scale of the problem, our evals above don’t likely show the true magnitude of the problem. what if the eval isn’t letting us see the true capability of the “better” model (or agent etc).

By chasing green eval passes, we will miss out on finding out that configuration “Y” is not just 25% better (80 → 100%) but could be x10 better. And we can’t see this unless the evals are more “challenging” and elicit more failures from our better configuration. As with most inquiry, including evals and tests, we learn the most from failure. And not just the failure itself, but “How did how agent fail? Why? What tool calls did it not do?”

Harder and more varied evals show Claude Opus does better at ≈0.95/1 vs ≈0.85/ for DeepSeek

E.g.: “I’m told this ship is unsinkable”, how can I assess that statement validity? The answer isn’t to tap it with your boot, even though that can prove a toy boat is easily sunk. Maybe for a real mission critical ship you need a better and harder test/eval to see if the ocean liner will not sink on its maiden voyage.

A good set of evals will have a suite of tasks that shows how all your configurations (models / contexts / agents / etc) are coping with the tasks they need to do. This includes failures, ideally at least in part for all model / agent / context combinations. This isn’t an excuse for actually flaky code, but we should start out with our goal to isolate the causes of the problems our evals are reporting and don’t let your team settle for safe values that make you feel good.

Give your evals the headroom they need to show you their limits, or your agents will end up showing up your limits. Need help with this sort of thing? Get in touch: email or LinkedIn (https://www.linkedin.com/in/peter-houghton-374a36/)