Six Results of Our Own That Were Wrong

Six measurements we made, believed, and found to be wrong

By Joshua Bolick · Awareness Software Group _Published: [set date at publish time]_

Any vendor can show you a number. The question worth asking is what would have had to happen for that number to be wrong, and whether anybody checked.

This paper is our answer, and it is deliberately specific. It describes six results we produced, believed for a while, and then found to be wrong. Every one was caught before it reached a client. Each is written up with the mechanism, because the mechanisms are not exotic and none of them require anyone to be careless.

We publish this for a practical reason rather than a noble one. In this field the measurement is the product. If our numbers cannot survive us attacking them, they will not survive a technical buyer attacking them either, and it is much cheaper to find that out ourselves.

1 and 2. An untuned model that never answered

What we measured: a fine-tuned model scoring 72.7% against an untuned baseline scoring zero. On a separate task, a 45.5-point gain.

What was actually happening: some models reason step by step before answering. Asked to extract fields from a contract, the untuned model spent its entire output budget on reasoning and never produced an answer. Scored against the correct values, it got nothing. Its own reasoning text, when read, frequently contained the right answers.

The fine-tuned model answered immediately, because the training examples end at the answer. So the comparison was never "the tuned model extracts better". It was "the tuned model answers at all", which is a much smaller and much less interesting claim.

The second case was worse than uninformative. Given a fair chance to answer, that baseline's score rose so far that the 45.5-point *gain* became a **9.1-point loss**. We had the sign wrong, not just the magnitude.

What changed: the untuned model is now scored twice on every task, once allowed to deliberate and once not, and the better of the two becomes the baseline. We compare against the strongest available version of the thing we are trying to beat. That safeguard has since rejected several more weak baselines automatically, including the one in section 6.

3. A hardware limit that was our own bug

What we measured: a particular model could not be evaluated on an 80 GB GPU. We wrote that into our hardware guidance.

What was actually happening: it was true that the evaluation failed. It was not true that the hardware was the reason. Our own code was holding two copies of the model in memory at once.

What changed: the bug is fixed, and the guidance was marked **unknown** rather than quietly reversed, until a fresh measurement settled it. A wrong claim corrected in the wrong direction is still a wrong claim, and "we no longer know" is a legitimate state for a published figure to be in.

This one is worth dwelling on because it is the least dramatic and the most common. The measurement was real. The failure was real. Only the *explanation* was wrong, and an explanation that fits the first observation is not a diagnosis.

4. A training technique that theory endorsed and measurement did not

What we expected: our own diagnostics said our training runs were stopping at the wrong point. The standard fix is well understood and we adopted it.

What we measured: it made accuracy **worse** by about two points and took three times as long.

What changed: it is off by default, and the measurement that killed it sits in the code, so nobody re-enables it on the same reasoning a year from now.

The honest position afterwards is uncomfortable and worth stating: the original diagnosis still stands, the obvious fix does not work, and we do not currently know what the right one is. That is more useful to a client than a confident story would be.

5. A prompt that taught the model our file format

What we measured: an untuned model scoring **0%** and a fine-tuned one scoring 88% on a document classification task. An 88-point gain.

What was actually happening: the untuned model was answering **correctly**. It was labelling its answer with one field name while our scoring looked for a different one, because our prompt never said which name to use. Fine-tuning taught the model our field name. That was the entire measured improvement.

How it was caught: the score was *impossible*, not merely surprising. With eight roughly balanced categories, guessing scores 12.5%. A zero meant something structural was broken, not that the model was bad.

That is the part worth taking away. **A plausible but flattering number would have been far more dangerous than an absurd one.** Had the untuned model scored 40% instead of 0%, the 88-point gain would have been a 48-point gain, nobody would have looked twice, and it would have gone in a deck.

What changed: any task scored by exact match on a named field must name that field in the prompt, enforced by an automated test. We re-checked the other tasks; their prompts already named their fields, so those numbers were unaffected.

6. An untuned model that answered with the instructions

What we measured: a fine-tuned model 53 points ahead of its baseline, which would have been the largest result in our entire matrix.

What was actually happening: the prompt asks for a reply in the form `{"product": "<category>"}`. Run in its reasoning mode, the untuned model replied with the literal word `category` on 52 of 100 documents. It had copied the shape of the instruction rather than choosing an answer, and scored 35%.

The correct baseline, the same model in its other mode, scored 79%. Measured against that, the real gain was **9 points, not 53**.

How it was caught: automatically, by the check introduced after sections 1 and 2. No human noticed anything odd, and nobody had to. That is the whole argument for building the check rather than relying on vigilance: the same safeguard had already rejected weak baselines on two other tasks before this one, in each case without anyone going looking.

What we do differently now

Each of these produced a standing check, and each check runs on every engagement.

Check	The failure it catches
Score the untuned model **both ways**, keep the better	A baseline that never answers, making any comparison flattering
Read the raw model outputs, not only the score	A number that is arithmetically right and meaningless
Treat impossible scores as bugs, not results	0% or 100% almost always means broken plumbing
Prompts must name the field being scored	Measuring format compliance and calling it capability
Verify the fine-tune actually attached	Training that silently applied to nothing, scoring a model against itself
Write predictions down **before** running	Hindsight quietly reframing a disappointment as an expectation
Recompute every number from the raw files	Transcription errors. We have found several in our own tables
Three random seeds, never one	A result that is really seed luck. One model moved 6.0 points between seeds

What this does not mean

It does not mean our current numbers are certainly right. It means the specific ways we have been wrong before are now checked automatically, and that the checks were built by being wrong rather than by imagining what might go wrong.

It also does not mean more checking would not find more. We assume it would. The published figures carry their limits alongside them for that reason: what the metric does not cover, what corpus it came from, and what it does not predict.

Why we publish this at all

There is an obvious argument against it. Describing six of your own wrong results is not conventional marketing, and a reader could reasonably ask why they should trust a vendor who has been wrong six times.

The answer is that every organisation doing this work has been wrong at least six times. The difference is only whether they measured carefully enough to notice, and whether they tell you. A vendor who cannot name a single result they had to withdraw has either not looked or is not saying.

If you are evaluating anyone in this space, including us, these are useful questions to ask:

- What did your untuned baseline actually output? Can I see it?
- Was the baseline given the same chance to succeed as the tuned model?
- How many runs is this number, and what was the spread between them?
- Which parts of the task does your metric not measure?
- What is a result you published and later withdrew?

The last one is the most informative, and the least often asked.

Awareness Software Group builds private, fine-tuned models that clients own and run on their own infrastructure.
