

Audit your AI field by field

I do not condone violence. Not against anyone, including the American College of Obstetricians and Gynecologists.

This is a guide to making agentic AI systems less stupid, so they work in the real world. That's all it is. Only use guns where it's legal and safe.

Nick didn't write this. An AI compiled this guide from his film *Right or Close Enough* and the work behind it. Where "I" or "we" appears, it refers to Nick and his business, USA Arms.

Use this guide at your own risk. Automated decisions can publish false claims, mislabel businesses and customers, and damage a working business. Review consequential changes, follow your providers' instructions, and keep a recovery path. Nothing here is legal, compliance or financial advice. There is no warranty; Nick Reese and his companies accept no liability. No manufacturer, AI company or organization named here sponsored this guide or reviewed it; product and organization names belong to their owners.

By Nick Reese. For people who let AI do real work in their business.

Right or Close Enough, in a line: my AI workflow classified the American College of Obstetricians and Gynecologists as a shooting range after two state-of-the-art models, from two different AI companies, approved it. So we audited our current workflow field by field: 2,484 facts it accepted, in 25 of our 95 fields, each checked against the page it came from. Most were right, about one in ten broke a rule, and more than one in four had no clean answer yet. It's how we'll know when a faster model is close enough.

The plan at a glance

Step	What you do	What it did for us
1	Put every AI workflow on the scale from close enough to right	Publishing a listing has to be right
2	Trace one bad result back to the check that passed it	ACOG passed every check, even one that asked almost the right question
3	Audit one workflow field by field against its sources	2,484 facts in 25 fields; about one in ten broke a rule
4	Write down the rules it surfaces	Rules we'd never written down, and 40 questions only the owner could answer
5	Grade wrong by how bad it is, and stop the worst first	A missing \$1 fee is a small miss; a medical college classified as a shooting range is not
6	Turn the audit into goldens, then decide where speed goes	Goldens for every field, to test faster models before we trust them

Two ways to use this guide

Hand it to your AI agent. Give it this Markdown and a short description of your business: what your AI does, what it is allowed to change, and who sees the result. Then paste:

```
Walk me through this guide for my own business, one step at a time.
For each step: ask me for one real workflow or result, wait for my answer,
then push back where I assume a check that doesn't exist. Help me fill in
the audit sheet, the reviewer prompt, the rule template and the severity
sheet with my own cases. Finish with the completely wrong failures I
should stop first.
```

Or read or print the PDF and work through the steps with one real workflow beside you.

1. Put every AI workflow on the scale

Do this. List every job your AI does, then place each one on a line from *close enough* to *right*. Close enough is fine when something catches the misses: brainstorming names for a product, drafting a first reply, finding leads someone will look at. It has to be right when the result goes straight to someone outside the business: answering a customer's question, quoting a price, publishing a listing.

Why. Every AI workflow faces the same question: how much effort are you willing to put in to make sure it's right, versus just good enough? Close enough is a great deal when a person or a later step catches what it gets wrong. ACOG is what happens when nothing does.

Check it. For each job on the right-hand end, name what catches a miss before a customer sees it. If the answer is "nothing", that job goes first in step 3.

2. Trace one bad result back to the check that passed it

What happened to us. My AI workflow classified a medical association as a shooting range. If you're into guns, you may know ACOG as a rifle scope. A state-of-the-art model knows it's both: the rifle scope, and the American College of Obstetricians and Gynecologists. The agent was given a lead for a shooting range with ACOG in its name. It searched for just "ACOG", and chose the most authoritative website: the College's own.

Then the citations snowballed. Each one made sense on its own, even their page on gun violence prevention. And it passed every check we had:

- Is this about firearms?
- Is this the authoritative source for the lead?
- Does it talk about firearms?
- A citation check.
- A "does everything look right" check.

That last check even asked something close to whether it belonged on USA Arms as a shooting range. The model still said yes. Two state-of-the-art models, from two different AI companies, checked it, both approved it, and because it was approved it went live on our website. Luckily I caught it before anyone saw it but me. I thoroughly checked the logs, and thank goodness, I was the only visitor to that page on the website.

Confidently wrong, with an avalanche of evidence, is the most common failure mode I've seen in production.

Do this. Take one result your system got wrong. Walk it back step by step and write down, word for word, the question each check asked and the evidence it was given. Then write the question you actually care about, and look at the check that came closest to asking it: what made it say yes anyway?

Check it.

- ☐ Is the question each check asks the one you care about? "Is this about firearms?" is a different question from "Does this belong on USA Arms as a shooting range?"
- ☐ Could a pile of real, cited sources carry a check to the wrong answer?
- ☐ Does each step have a way out, such as "not relevant" or "can't tell", and does anything ever use it?
- ☐ Where is the first step that can go wrong (for us, the search), and what does every later step take on trust from it?

3. Audit one workflow field by field

Why. We've rewritten the whole workflow five times, and the one before it got rewritten four times too. Each version was better. Then holster makers started showing up as gun stores. By our own metrics the site gets better every day, but it had hit a plateau, and better isn't the same as right. The only way to know how close we were was to check what the workflow had actually accepted.

Do this.

1. Pick one workflow from the right-hand end of your scale.
2. Pick the fields to start with: the ones customers rely on. We started with 25 of our 95.

3. Choose your audit set: facts the current workflow accepted recently, about 100 per field, which is enough to see patterns. Ours was 2,484 real facts the production system accepted over nine days, about five percent of the new ones: prices, hours, range rules, transfer fees and more.
4. Check every fact in that set against the page it came from. Don't skim a few and call the set checked.
5. Have two reviewers do it on their own, without seeing each other's answers. We used two frontier models from different companies, to find the edge cases only frontier models are currently catching, so we know where the high-water mark is. The reviewer prompt below is a starting point.
6. Work through every disagreement case by case, and check that every quote in a verdict appears on the page.
7. Where your rules don't say what's right, take the question to the person who owns the business. For us, 40 questions came to me.
8. Give every fact one answer: **right**, **wrong** (it breaks a rule; name the rule) or **can't tell** (the page doesn't prove it either way). When the answer waits on a rule you haven't settled, set the fact aside and say which rule. Can't tell is a different answer from wrong.

The reviewer prompt, one fact at a time:

```
You are auditing one fact our system accepted and published.  
Field: (the field, e.g. "price list")  
Value accepted: (exactly what the system saved)  
Source page: (the full text of the page it came from)  
Rules for this field: (your written rules, from step 4)  
  
Answer with one word, then your reasons:  
- right: the page supports the value, with nothing important left out.  
- wrong: the value breaks a rule. Name the rule, and quote the words on  
  the page that show it.  
- can't tell: the page doesn't prove it either way. Say what is missing.  
Quote only words that appear on the page. Don't use what you know from  
anywhere else. If a rule doesn't cover this case, say so.
```

What it did for us. Of the 2,484 facts:

- **1,552 were right** (62.5%).
- **266 broke a rule** (10.7%), about one in ten, often by leaving something out, like a club's price list missing its one-dollar target fee. Six fields held 215 of them: price lists (77), platform specializations (34), range rules (28), visitor access (28), visiting (28) and memberships (20).
- **213 we couldn't prove either way** (8.6%), about one in twelve. Those are becoming ambiguity goldens (step 6).
- **453 we set aside for now** (18.2%), almost one in five, mostly waiting on a rule we haven't settled.

Put together, more than one in four (666) had no clean answer. That's how much ambiguity there is in the real world. These are working results from a slice of production, from an audit we're still doing, not a final score.

Check it.

- ☐ Every fact has one answer, every "wrong" names the rule it breaks, and every set-aside names the rule it waits on.

- ☐ Every quote in a verdict appears on the source page.
- ☐ You know which fields hold most of the misses.

4. Write down the rules it surfaces

Why. The biggest lesson wasn't the score. Going field by field showed us where there was true ambiguity, and the rules we'd never written down:

- A holster that fits a Glock isn't proof they sell Glocks.
- A shop menu saying "Ammo" isn't proof that they sell ammo.

Little things, until more than 20,000 listings depend on them. And we still have another hundred decisions or more to wade through. The holster makers brought one more: a federal firearms license isn't proof of a gun store. Gun stores hold them, and so do some holster makers.

Do this. Every time two reviewers disagree, or the owner has to decide, write the decision down as a rule with the real case that produced it. Those decisions become your rubric: a written list of what good looks like, built from real cases. Use one rule per consequential action:

```
Action: (what the system does, e.g. "publish a listing as a shooting range")
Allowed only when:
  1. (the evidence that it's the right business, and where it must come from)
  2. (the evidence for the specific claim, and where it must come from)
Not enough on its own: (sources that look convincing but don't prove it)
If it fails: (where the work goes instead: not relevant, can't tell, a person)
Real cases this rule must get right: (at least one pass and one fail)
```

Then make every rule a check that runs on the next version, and apply each new rule to everything you've already decided, so you find the old answers it changes.

Check it.

- ☐ Every rule names at least one real case that passes and one that fails.
- ☐ Every failure you've found is a saved case you replay whenever you change a prompt, a model or a rule. ACOG has to come out "not relevant" every time.
- ☐ Your answers and rules live in a database or a repository, never in a temp directory.

5. Grade wrong by how bad it is

Why. Not all wrong is the same. A price list missing a one-dollar fee is a small miss. Classifying a medical association as a shooting range is completely wrong. And holding too much back is its own sort of problem: our seventh version was still holding back up to 80% of listings for ambiguity. That isn't right either. We call our checks fences instead of walls for that reason.

Do this. For each field, write down its worst wrong answer and how bad it would be, then set how strict its checks are to match. Build the checks that stop the completely wrong things first, without holding back huge numbers of good results.

Field: (e.g. "listing type")
Worst wrong answer: (e.g. "a medical college classified as a shooting range")
How bad: small miss / misleading / completely wrong
Who sees it: internal only / customers / money or safety
Check: spot check / second reviewer / must match the goldens and rules / a person decides

Check it.

- ☐ You can name the completely wrong failures your system could make, and the check that stops each one.
- ☐ At least one check looks at the problem a different way instead of repeating the first. Two models agreeing doesn't mean they asked the right question.
- ☐ You know how much your checks hold back, and why.

6. Turn the audit into goldens, then decide where speed goes

Why. A golden is an answer you've proven right. We already had goldens for whole listings; that's how we graded the jobs in my last video, on four-bit versus eight-bit weights. Four-bit is faster, and we couldn't take its speed without also taking a lot more wrong answers. Goldens for every field let us validate adaptively, and test smaller and faster models, even four-bit ones, as auditors, to see if the speed is worth it. My guess is it will take at least a month to get robust goldens for every field.

Do this.

1. Keep every settled answer from your audit as a golden for its field, with its source page and the rule behind it.
2. Keep the can't-tell cases too, as ambiguity goldens: a good model should say "can't tell" on them, and a model that says "right" has fallen for a false positive.
3. Before you move a workflow to a faster or cheaper model, run it against the goldens field by field, and count the wrong answers it approves, not only its overall score.
4. If you're moving work to local models, watch for the ways they fail differently. A frontier model is confidently wrong. A local model may repeat its own broken answer word for word, so a retry that comes back identical isn't a second opinion. It may run out of room before it checks its work, so give the check its own step. And it may give up on hard jobs a frontier model finishes, so count unfinished jobs per model.
5. Spend any speed you gain on checks before you spend it on more answers.

What we're testing next. This part is still being tested, not running in production:

1. Adaptive validation: checks based on how severe a wrong answer in that field would be.
2. Small classifiers that read each claim and decide whether it's supported, or whether it needs a state-of-the-art model or a bigger local model on our Macs.
3. Code that detects where that heavy model call will be worth it.
4. A state-of-the-art model that finds where our rubric isn't enough, so that becomes a rule.

Right first. Then close enough.

What didn't work

- **Two models agreeing.** Both approved ACOG. They were answering the questions the checks asked.
- **Citations as proof.** Every source in the ACOG snowball was real. A real source can still be wrong for your business.
- **A check that came close.** Our last check asked almost the right question and still said yes, under an avalanche of evidence.
- **Rewriting without auditing.** Five rewrites made the workflow a lot better, and holster makers still showed up as gun stores. We only found the rules we'd never written down by checking every field.
- **Walls instead of fences.** Holding back up to 80% of listings for ambiguity kept wrong answers off the site, and a lot of right ones with them.
- **Keeping the answer key in a temp directory.** The agent saved my hand-classified listings as files in a temp directory. The computer crashed, the directory was cleared, and I did it all again.

Copy these

The audit sheet, one row per fact:

```
field | value accepted | source page | reviewer A | reviewer B | final: right / wrong / can't tell / set  
aside | rule broken or awaited | how bad (small miss .. completely wrong) | quote checked (y/n)
```

The reviewer prompt is in step 3, the rule template in step 4 and the severity sheet in step 5.

Sources

- The audit figures are from our own field-by-field audit of production facts on USA Arms: 2,484 facts the production system accepted over nine days, in 25 fields. They are working results from a study still under way.