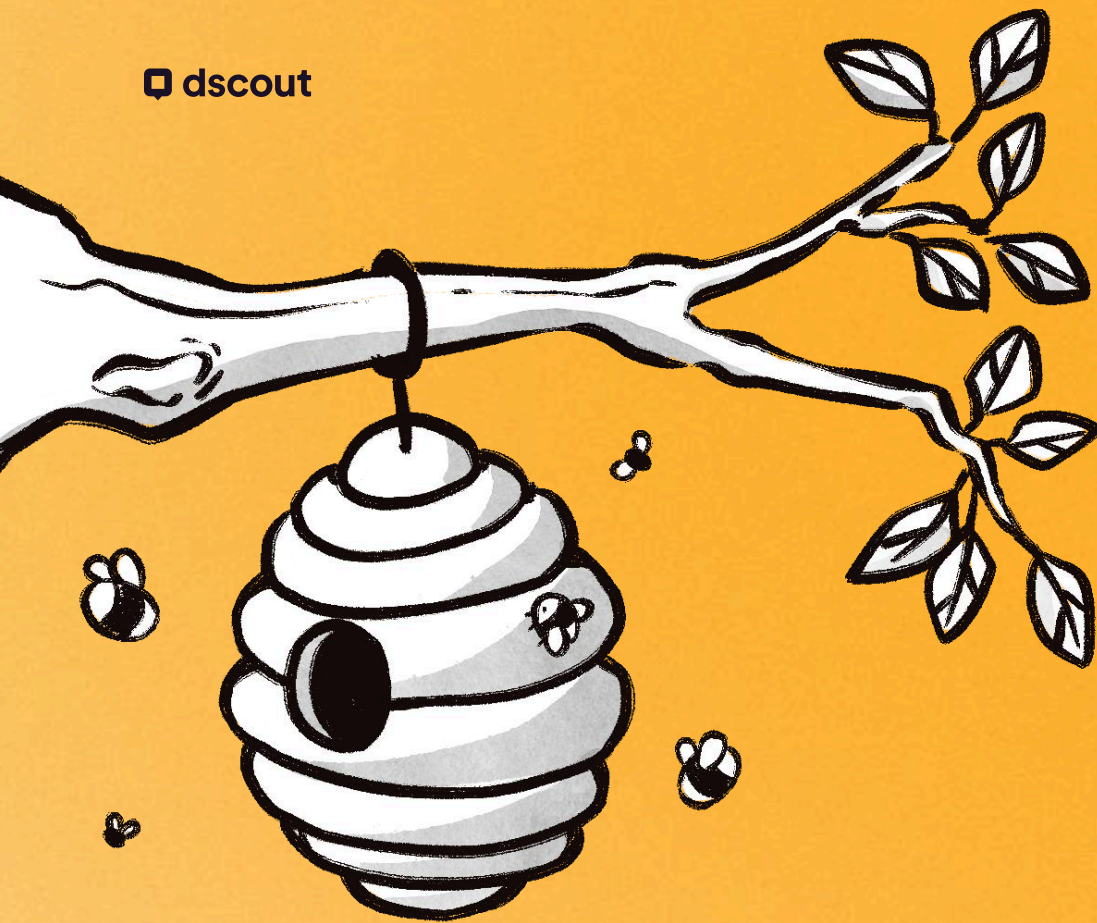




The People Nerds Guide to AI Evaluations

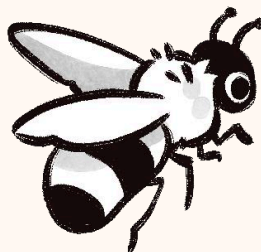
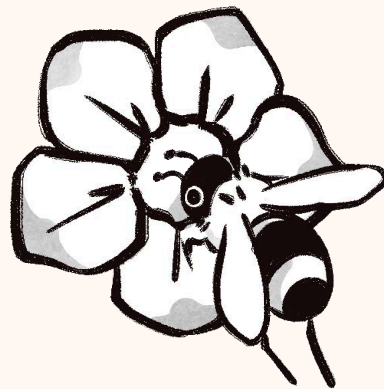
A UX-minded approach to evaluating AI features, improving outputs, and shaping product quality

By Elizabeth (EA) Allen, Senior Product Manager at Dscout



Jump *to*:

- ³ Intro to AI evals: How to keep your hive humming
- ⁴ Your guide to building AI evals
 - ⁵ Where to start
 - ⁷ What to evaluate
 - ⁹ Example from our own work: Refine Your Questions
- ¹¹ What to do with what you find
- ¹⁴ UX teams, this is your moment!



INTRO TO AI EVALS:

How to keep your hive humming

Learning how to evaluate your product's AI functions may sometimes feel like grasping in the dark for the right solutions. But there is both an art and a science to honing the process, and it all starts with evals.

For example, when my team at Dscout first worked on our [Refine Your Questions](#) feature, we noticed something weird. The AI was flagging research questions as “double-barreled” (a no-go for research questions!) that really weren't—things like “Do you think super-shuffle is worth paying for? Why or why ” That's not two questions. It's one question with a bit of framing. But the system didn't know that.

That surprise kicked off a conversation about what else the feature might be getting wrong, and how we could check more systematically. In other words: **we needed evaluations.**

We needed a process to systematically check whether our AI features were actually doing what we want it to. But where to begin?

If you've ever watched a beekeeper at work, there's a rhythm to it. The hive hums along on its own, doing what bees do. Every so often, the beekeeper opens it up—is the queen healthy? Are the workers doing their part? Anything weird that needs attention?

That's what eval work is for AI features. The thing runs on its own out in the world. You open it up periodically to see if everything's actually healthy, so you can catch issues before they get bigger.

In this guide, we'll walk through how to build AI evals and assess if your AI tools are doing what you want them to do.

Your guide to building AI evals

AI evals—short for evaluations—are a way to systematically check whether an AI feature is actually doing what you want it to.

If you’ve built a customer service chatbot, for example, you’d want to review its responses to make sure they’re helpful, accurate, and on brand. If you’ve ever spot-checked the output of an AI feature and thought, “Hmm, that’s not quite right,” congratulations! You’ve already been doing vibe evals.

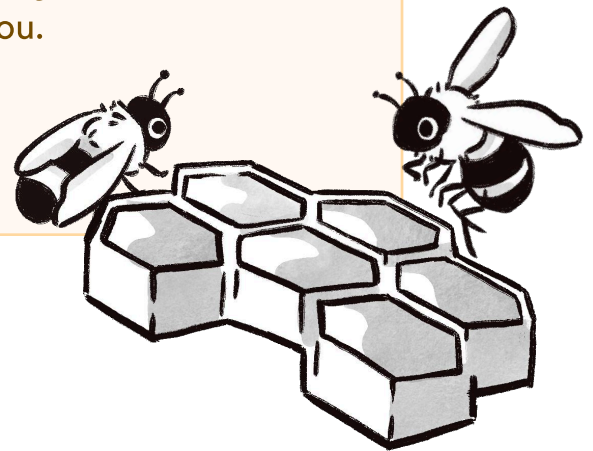
Human evals include those vibe checks, but they also include working through data more systematically—having a human review each output, label it as good, bad, or unclear, and jot down notes. You can do this in a dedicated tool like [Langsmith](#) or [Braintrust](#), or just in a spreadsheet.

Automated evals are when you write out a checklist (sometimes called a scorer) and have another LLM (large language model) use it to score the outputs for you.

HEADS UP!

It’s tempting to jump straight to automation. But here’s the thing: humans have to define what “good” looks like before a machine can spot it for you.

That’s why we start with human evals.



Where to start with AI evals

AI evals can seem daunting. AI features can do all sorts of things, so it's important before you start setting up the process to make sure your ducks are in a row with your team and your data.

✓ Assemble your team

You need two people at the table: someone who knows what “good” looks like (your subject matter expert, or SME) and an engineer.

Here's the thing: for a lot of AI features in product work, the SME is you. If you're a researcher or designer, you already have the taste needed to determine what a well-written research question looks like, what a helpful support response sounds like, what a reasonable user flow looks like. That muscle is exactly what eval work needs.

The engineer's job is the plumbing:

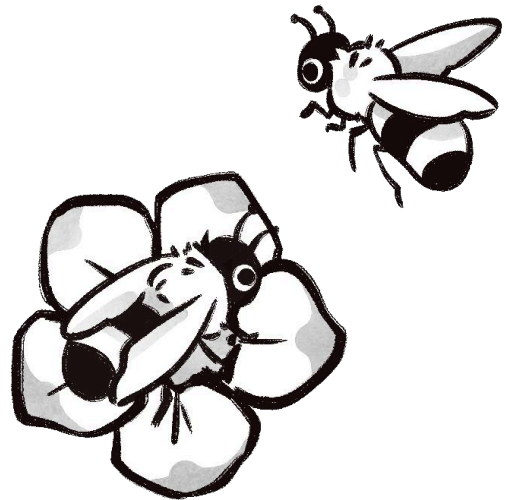
- Setting up the dataset
- Getting it into a tool where you can evaluate it
- Helping you write the evaluation prompt (making sure the checklist is actually looking at the right pieces of each data point)
- Making sure results come back in a format you can read and review

Heads up: this takes several tries. Be patient and prepared for a lot of back and forth.

✓ Assemble your data

Sticking with the customer support chatbot example, what would you want to look at? That may include data like...

- The full session
- Each user query
- Each bot response
- Whether the user rated the conversation thumbs up or thumbs down
- User ID
- Account ID
- Chat start time



To start, this might be test data you or your team made up (sometimes called synthetic data). But the best value comes from real customer conversations, which means looping in your privacy and security counterparts. Figure out what you can use and how to adequately protect customer data. If your feature hasn't shipped yet and you don't have customers using it, synthetic data is how you get going.

How much data should you start with? Good news: you don't need thousands. You can get meaningful signals from 20 to 50 solid examples. It's actually preferable to start small because it lets you quickly review outputs and set things up.

One more thing: your dataset should be diverse. Happy path scenarios, things gone obviously wrong, and weird edge cases—all of it. The edge cases are often where evals earn their keep.

What to evaluate

Once you have data, the next question is: what do you actually want to check? Look at it through a user experience lens. If you're evaluating a customer support chatbot, things like:

- Is the person's problem getting resolved?
- Is the bot following brand guidelines?
- Are things being correctly escalated to a human when the issue is beyond what a bot should handle?
- How do we prevent customer/user frustration?
- What's the thing that, if it went wrong, you'd most want to catch?
(For example: the bot hallucinates an important policy.)

Notice how these are the same questions you'd ask in any usability study. That's not a coincidence. Eval work is usability work in a different shape!

Those questions span a few different dimensions:

- Accuracy
- Tone
- How edge cases are handled
- How the experience feels
- Safety

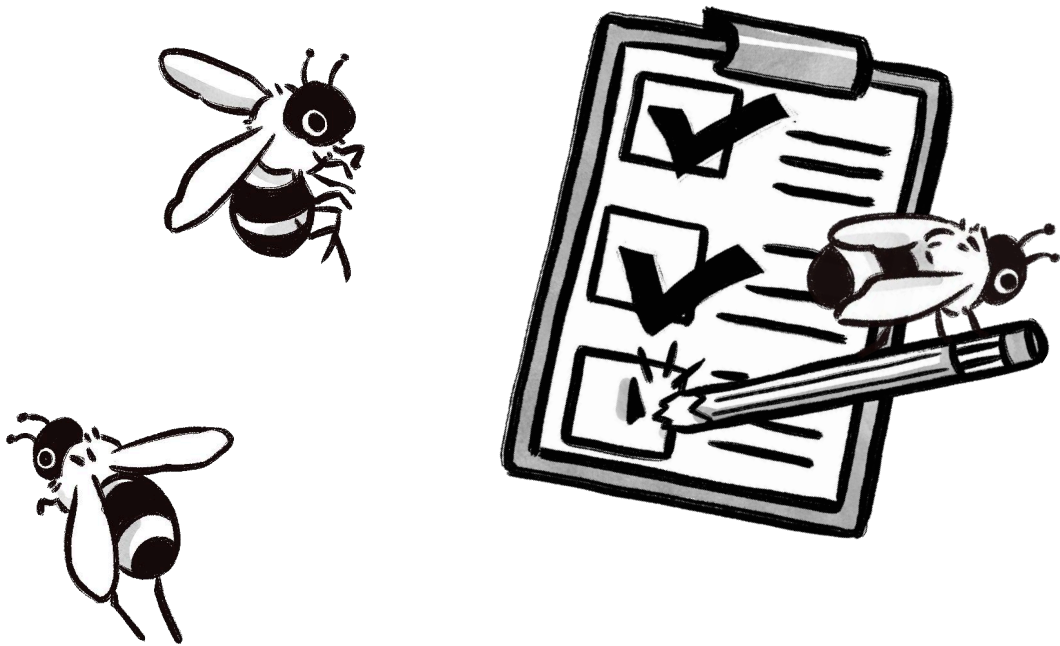
You don't have to evaluate all of them at once. Pick the ones that matter most to your feature.

✓ Start with vibes

Before you write any formal checklist, just spend time with the outputs. Read them. Notice things. Form opinions. This is the same muscle a researcher uses when they sit with interview transcripts before deciding what's worth tagging. You're not tagging yet, you're getting a feel for the material.

That vibes work is what generates the raw material for your checklist later. Which is why I'd steer you away from out-of-the-box scorers. They skip the part where you figure out what good looks like in your context. And if someone asks you why a response scored 75% on "helpfulness," you'll have a hard time answering. What does a 75% helpfulness score even mean?

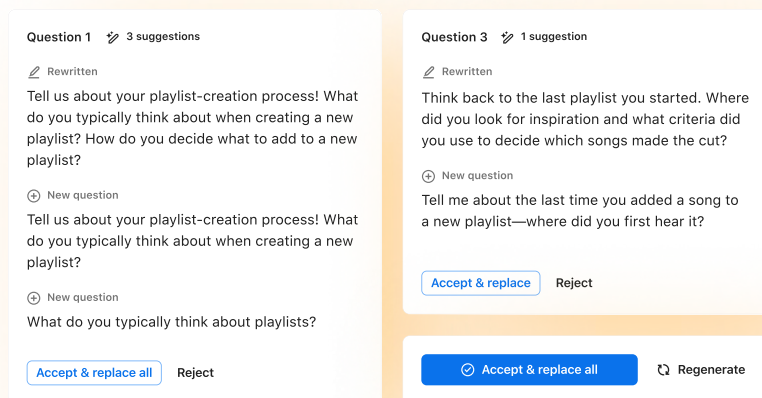
Your scores need to be defensible. Writing your own scorer is how you define what good is—the same promise we made at the top of this article.



Example from our own work: Refine Your Questions

✓ The feature

Refine Your Questions is a Dscout feature that offers suggestions for how to reword research questions for clarity. It catches confusing terms and double-barreled questions.



✓ The challenge

Let's stick with the double-barreled example since it's easy to follow. A classic double-barreled question stuffs two actual questions into one, and the person answering can't give a clean response to either.

But the feature was flagging things that weren't really double-barreled. "Do you think super-shuffle is worth paying for? Why or why not?" isn't two questions. The "why or why not?" is just framing—an invitation for the person to explain.

An engineer looking at structure alone ("two question marks = double-barreled") wouldn't catch that nuance. That's exactly where research fluency does real work. A researcher on my team helped me gut-check these examples, which is the kind of domain judgment that makes evals better.

✓ The prompt

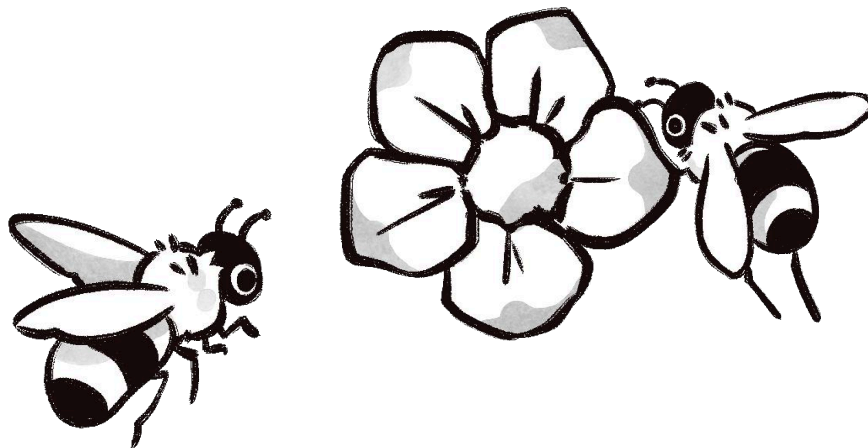
Once we had the pattern, we updated the prompt. Instead of just looking at structure, the system now considers the context of the second question: does it provide framing or structure for the first? If so, no flag needed.

Then we re-ran evaluations on the same dataset. Two things to check, not just one. First, the questions that were incorrectly flagged (first flagged by AI, then checked by a human—good!). Second, genuine double-barreled questions still got caught (also good). If you skip that second check, you can fix one problem and quietly create another. Make both directions part of your routine.

✓ The takeaways

Was this feature high-stakes? Not necessarily, we're constantly building AI features that are more robust than Refine Your Questions. But even a smaller, more low-stakes feature deserves eval work. If we consistently got something this simple wrong, we'd lose user trust—or researchers would end up with unnecessarily long studies full of duplicative questions. The trust cost of small misses adds up.

The bigger lesson, worth taking to your own work: evals are where you catch your AI being too rigid, matching on surface patterns but missing nuance a human would intuitively understand. That's true whether you're working on a research tool, a chatbot, or anything else.



What to do with what you find

Once you start seeing patterns in your eval results, you've got options.

Findings usually point to one of a handful of changes:

- Your prompt needs updating.
- Your eval itself is too loose or too strict.
(Sometimes the finding is about the eval, not the AI.)
- Something needs to change in post-processing—the steps that happen to the AI's output before the user sees it (filtering, reformatting, adding disclaimers, and so on).
- You need a separate eval for a different case. In our Refine Your Questions work, for example, we realized task-style questions needed different treatment. We run the clarity check but skip the double-barrel check for those, because task questions commonly bundle several prompts on purpose.
- You need to go back to the drawing board entirely.

Not every eval failure needs a fix. Some are edge cases you can accept, some expose a deeper problem, and some just aren't worth the engineering cost right now. These are calls to make with your team.

Weigh the visibility of the feature, the risk of getting it wrong, and the time and resources you have. Refine Your Questions isn't perfect (as most AI tools are not!), but because it's relatively low-risk, we felt comfortable releasing it.

✓ Use answers for better input experience

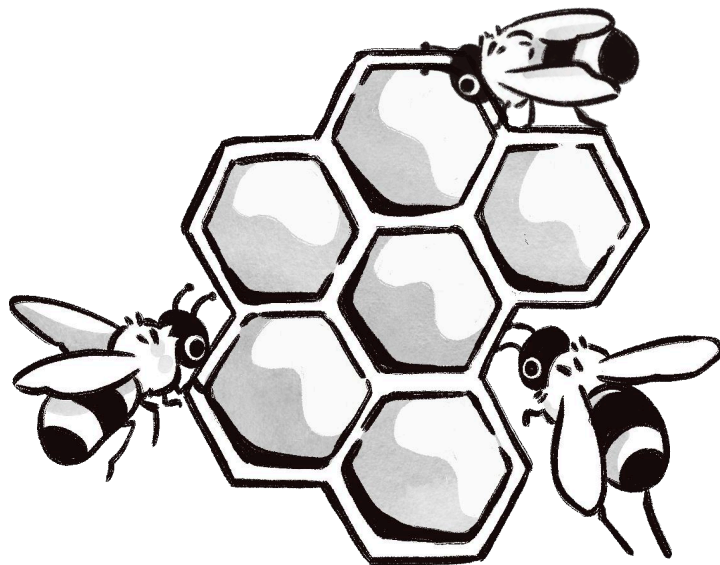
Evals can highlight that users aren't giving great inputs, but you don't always need an eval to spot that risk. Sometimes you can see it coming and design for it from the start.

Take [Explore Your Data](#), our feature that lets researchers ask questions of their data in natural language. We knew vague or open-ended inputs would lead to less useful outputs, so we added prompt starters—short, concrete examples that model what a good question looks like. Things like:

- Wins – Identify moments of delight in the study
- TLDR – Generate a concise bullet TLDR of responses

Each one shows users what they can ask, the shape a useful question takes, and the kind of result they should expect. We didn't need an eval to tell us this—we used UX instinct. But the outcome is the same one good eval work can lead to: an experience where the AI gets better inputs, so users get better outputs.

Sometimes the answer is more AI. Sometimes it's just a more thoughtful user experience.



Playing the long game

From my experience, getting your immediate pod engaged with eval work is harder than I expected! This work is new, time consuming, and it's a muscle that a ton of teams are still building.

To help you get started, here are a few things that I've found really helpful.

✓ Bring evals into the rhythm your team already has

Introduce AI evals into some common processes. For example...

- Drop a low-scoring output into morning standups so people see what users are actually getting.
- Bring eval findings into design crit. Let your designers and researchers weigh in on what's working and what isn't.
- Pull specific eval examples into product debates, especially when there's disagreement about what "good" looks like.

✓ Bring your judgment and thoughts

The shape of the conversation matters, too. Don't just share the score, share the judgment behind it.

“““

When you bring teammates into your thinking—why you scored something the way you did, what felt off about it—you're inviting them to define what good looks like alongside you.

That's where the most fruitful conversations happen.

To help make this easier, I'm building an automation that pipes low-scoring examples into a Slack channel so the team sees them every morning. The lower the friction, the more this work compounds.

These conversations don't have to be dramatic to be valuable. Take an example from our work on Explore Your Data: we were getting quotes back with a lot of filler—"umm, hmm, yeah, uh-huh"—that weren't useful for anyone. Small finding, easy fix. We updated the prompt to strip those out, and now quotes get right to the point. Not every eval moment is a crisis. A lot of them are quiet, quality wins.

UX teams, this is your moment!

One huge payoff is when eval work becomes a foundation for cross-functional alignment on quality. Once your pod is in the habit of looking at outputs together, you have a shared language for what good means—which makes every product debate downstream sharper.

For UXRs and designers in particular, [AI evaluation work changes your seat at the table.](#)

You're not the person flagging concerns from the sidelines; you're the one holding the data, driving the quality conversation, and making decisions defensible. That's a different kind of influence than research and design folks usually get on AI work, and a place where your taste is invaluable.

You don't need to be technical to do this work. You need to know what good looks like—and you already do.

So pick one feature. Gather 20 examples. Get this process started. You've got this.





Dscout helps you build with confidence.

Learn how our AI-powered tools give you a faster path to understanding and allow the whole team to run high-quality end-to-end research with ease.

See our tools in action

