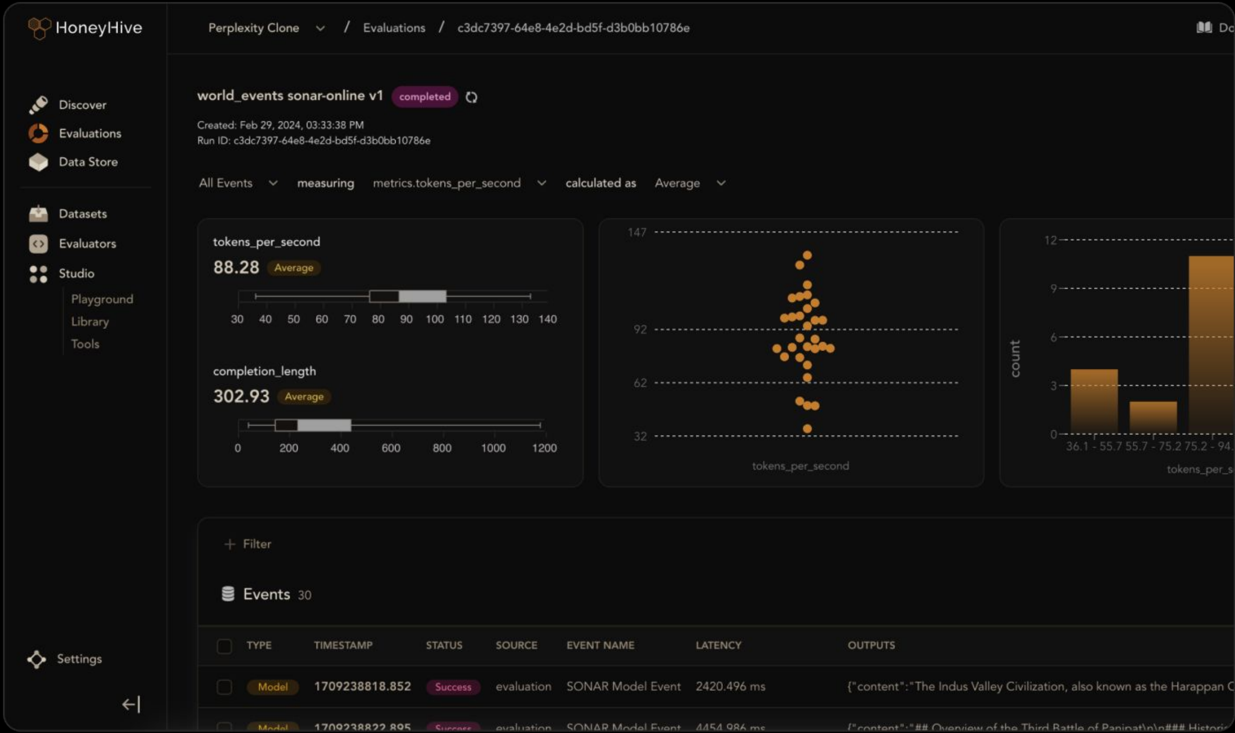
HoneyHive

Evaluating AI Applications



world_events sonar-online v1 completed

Created: Feb 29, 2024, 03:33:38 PM
Run ID: c3dc7397-64e8-4e2d-bd5f-d3b0bb10786e

All Events measuring metrics.tokens_per_second calculated as Average

tokens_per_second
88.28 Average

30 40 50 60 70 80 90 100 110 120 130 140

completion_length
302.93 Average

0 200 400 600 800 1000 1200

Events 30

	TYPE	TIMESTAMP	STATUS	SOURCE	EVENT NAME	LATENCY	OUTPUTS
☐	Model	1709238818.852	Success	evaluation	SONAR Model Event	2420.496 ms	{"content":"The Indus Valley Civilization, also known as the Harappan C
☐	Model	1709238822.895	Success	evaluation	SONAR Model Event	4454.986 ms	{"content":"## Overview of the Third Battle of Panipat\n\n### Histori

HoneyHive

About me

Worked on GenAI at Microsoft starting 2021

Co-founded HoneyHive in late 2022 to build tooling for AI applications

At HoneyHive

- Worked with multiple teams on architecting AI evaluation systems
- Conversed with all major builders & researchers in the space
  - Application layer, model layer, infra layer
  - Multi-agent systems, Text-2-SQL, Financial QA, AI Search Engines, etc. - from Fortune 50 to pre-seed startups

HoneyHive

Two simple questions to cover today

Can I trust my AI application?

How do I build that trust?

+ 1 practical example of an evaluation system

HoneyHive

How will we get to this?

Can I trust my application?

- How do I assess trust? What’s evaluation?
- How did people in the past (DevOps/MLOps) evaluate systems?
- How do those ideas translate to GenAI? Why is it tricky?
- How do I measure performance in that context?

How do I build trust?

- Based on these past insights, how do I build a repeatable eval system today?
- An example evaluation system for a multi-agent system

HoneyHive

What could go wrong with AI?

Bard/Perplexity/You.com lies to you

Customer support bot makes false promises to customers

Financial QA bot provides the wrong insight at a critical moment

Data extractor misses key features in the unstructured data

Trust is very hard to rebuild once broken.

HoneyHive

To assess trustworthiness, we evaluate

Will my application perform well in front of my users?

When it doesn’t, can I tell what went wrong?

Can I stop it from doing something bad in the first place?

HoneyHive

How is this done?

To assess how well it will perform

- See how your application performs in a close simulation of production

To assess what went wrong

- Find which part of your application didn’t behave as expected

To stop something bad from happening

- Continuously test your application while building
- Find a tell-tale sign of something going wrong live & stop it

HoneyHive

How was this done in the past?

	Will it do well?	What went wrong?	Can I stop it from failing?
Software	User Acceptance Testing	Regression Analysis Unit tests	E2E tests Integration tests Exception handling
Classical ML	Offline batch evaluation	Embedding drift Shapley values	Validation loss Live guardrails

HoneyHive

How to think about these ways of evaluation?

3 pieces of an eval

What are you evaluating - aka configuration / variant

What are you evaluating against - aka dataset

What are you measuring - aka evaluators

Putting it together

- We evaluate a *configuration* against a *dataset* using *evaluators*

HoneyHive

Each stage requires different type of eval

	Configuration	Dataset	Evaluator	Depth / Speed
Unit Test	Small component	Few expected inputs-responses	Crash / Output Regex	Low / Fast
Integration Test	Multiple components	Many expected inputs-responses	Crash / Output Regex	High / Slow
Offline batch evaluation	Production model	Many inputs-responses from production	F1 / Recall	High / Slow
Live guardrail	Live application	Live inputs	Output classifiers	Low / Fast

HoneyHive

How does this translate to GenAI?

The overall evaluation framework translates over

1. Run light evaluations when iterating
    - As I change things in one place, are things elsewhere not breaking secretly?
    - Unit tests, regression checks

2. Run heavy evaluations before deploying
    - Is my application capturing the distribution in production accurately?
    - Offline batch evaluations, integration tests, feedback from user acceptance testing

3. Setup fast guardrails to catch outliers live
    - Is my application about to do something horrible?
    - Live guardrails

But in the real world…

HoneyHive

The problem is evaluators

3 acceptable summaries for OpenAI's InstructGPT launch

- *OpenAI's InstructGPT, trained with human feedback, outperforms GPT-3 by following instructions more accurately for a safer user experience.*
- *InstructGPT surpasses GPT-3 using RLHF, improving instruction adherence, truthfulness, and reducing toxicity.*
- *InstructGPT offers businesses and developers an ethically aligned, more precise AI tool by outdoing GPT-3 in understanding user instructions.*

No fixed ground truth to count on anymore

- **F1, Recall, Eval loss, Output Regex are out of the window.**

1 unacceptable summary that doesn't throw an exception

- InstructGPT, offered by Anthropic, uses AI feedback to outperform GPT-3.

No deterministic failure modes to count on

- **Exception handling is out the window.**

This is why GenAI evaluation is hard

Hard to say if something went well

- Subjectivity
- Lack of ground truth
- Non-determinism

Hard to say what went wrong

- Complex apps have multiple points of failure (ex: RAG)

Hard to stop something dangerous

- Humans can’t review everything in production (ex: Devin)

What’s the common piece?

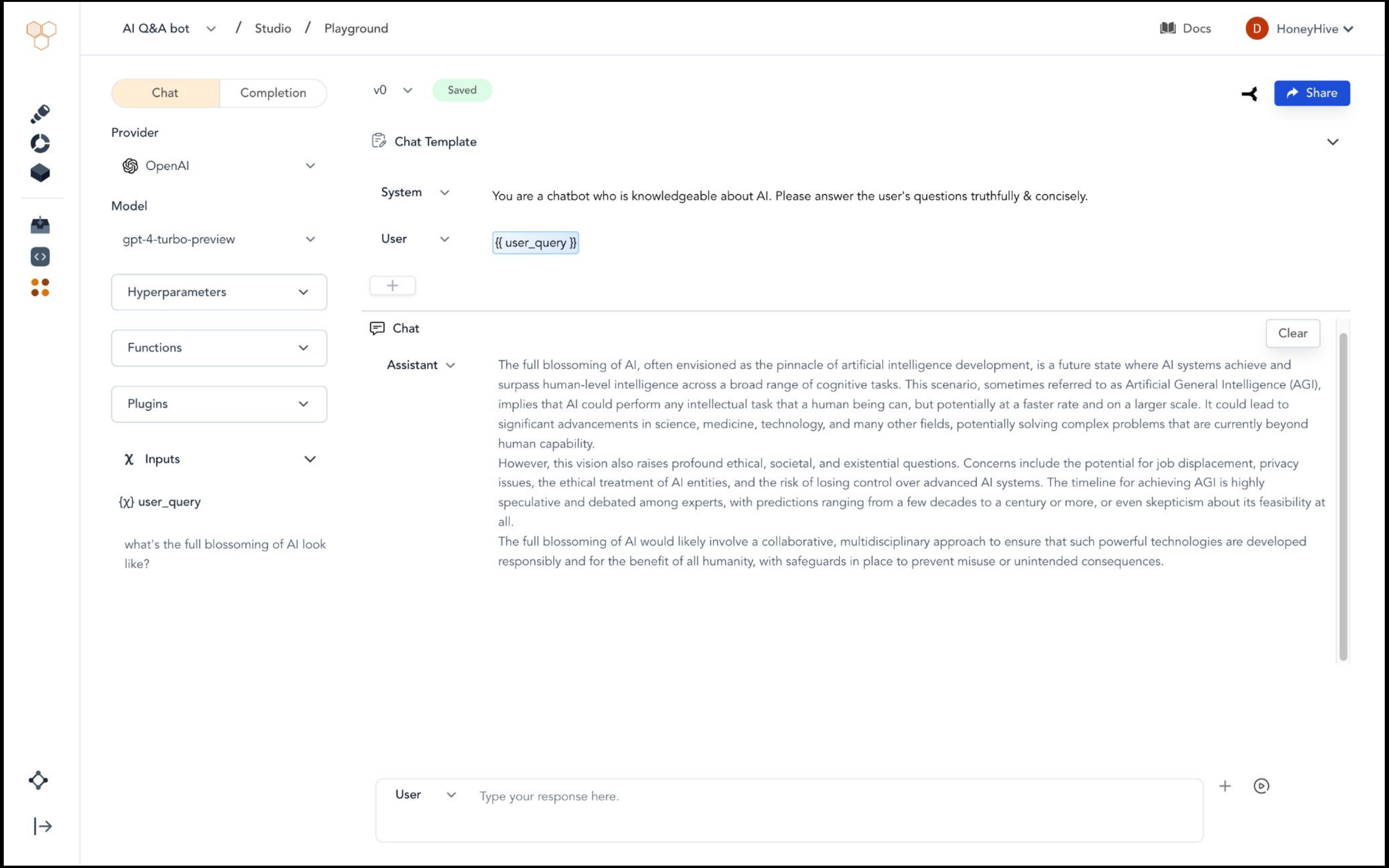
An intelligent application requires a more intelligent evaluator to evaluate quality

Human evaluation is very important

1. Humans can take subjectivity into account
2. They can review complex traces to root cause issues
3. Can listen to user surveys to understand common failure modes

HoneyHive

"vibe check"



AI Q&A bot / Studio / Playground

Docs HoneyHive

Chat Completion

v0 Saved

Share

Provider: OpenAI

Model: gpt-4-turbo-preview

Hyperparameters

Functions

Plugins

Inputs

{χ} user_query

what's the full blossoming of AI look like?

Chat Template

System You are a chatbot who is knowledgeable about AI. Please answer the user's questions truthfully & concisely.

User {{ user_query }}

Chat

Clear

Assistant The full blossoming of AI, often envisioned as the pinnacle of artificial intelligence development, is a future state where AI systems achieve and surpass human-level intelligence across a broad range of cognitive tasks. This scenario, sometimes referred to as Artificial General Intelligence (AGI), implies that AI could perform any intellectual task that a human being can, but potentially at a faster rate and on a larger scale. It could lead to significant advancements in science, medicine, technology, and many other fields, potentially solving complex problems that are currently beyond human capability.

However, this vision also raises profound ethical, societal, and existential questions. Concerns include the potential for job displacement, privacy issues, the ethical treatment of AI entities, and the risk of losing control over advanced AI systems. The timeline for achieving AGI is highly speculative and debated among experts, with predictions ranging from a few decades to a century or more, or even skepticism about its feasibility at all.

The full blossoming of AI would likely involve a collaborative, multidisciplinary approach to ensure that such powerful technologies are developed responsibly and for the benefit of all humanity, with safeguards in place to prevent misuse or unintended consequences.

User Type your response here.

HoneyHive

How to harness “vibe check” effectively?

The mantra is: **minimize cognitive burden**

1. Leverage simple judgement criteria
   - Binary ratings (good/bad)
   - Likert scale (how good on scale of 1-5)
   - Rankings (which one’s better/worse)
   - Implicit feedback (rage clicks/insults/UI interactions)
2. Surface only relevant examples
   - Highlight regressions
   - Select only few from different topics
3. **Codify heuristics**
   - Codify “gut instincts” into simple if-then rules, also called “assertions”
   - Assertions can be included in unit tests & live guardrails

HoneyHive

Is this enough?

HoneyHive's mission is 6-sigma reliability for AI applications.

- 6-sigma means 3 failures per million runs.

Using basic statistics, this means testing against ~800k examples.

Alpaca Eval found that MTurk evaluators

- costed $300 per 1000 ratings
- took 10 hrs per 1000 ratings
- had 60% consistency on ratings

So, to do such a deep evaluation, will take **8000 hrs & $240,000 for humans on MTurk** and an even higher price on Scale and Surge.

HoneyHive

What’s the alternative?

LLMs?

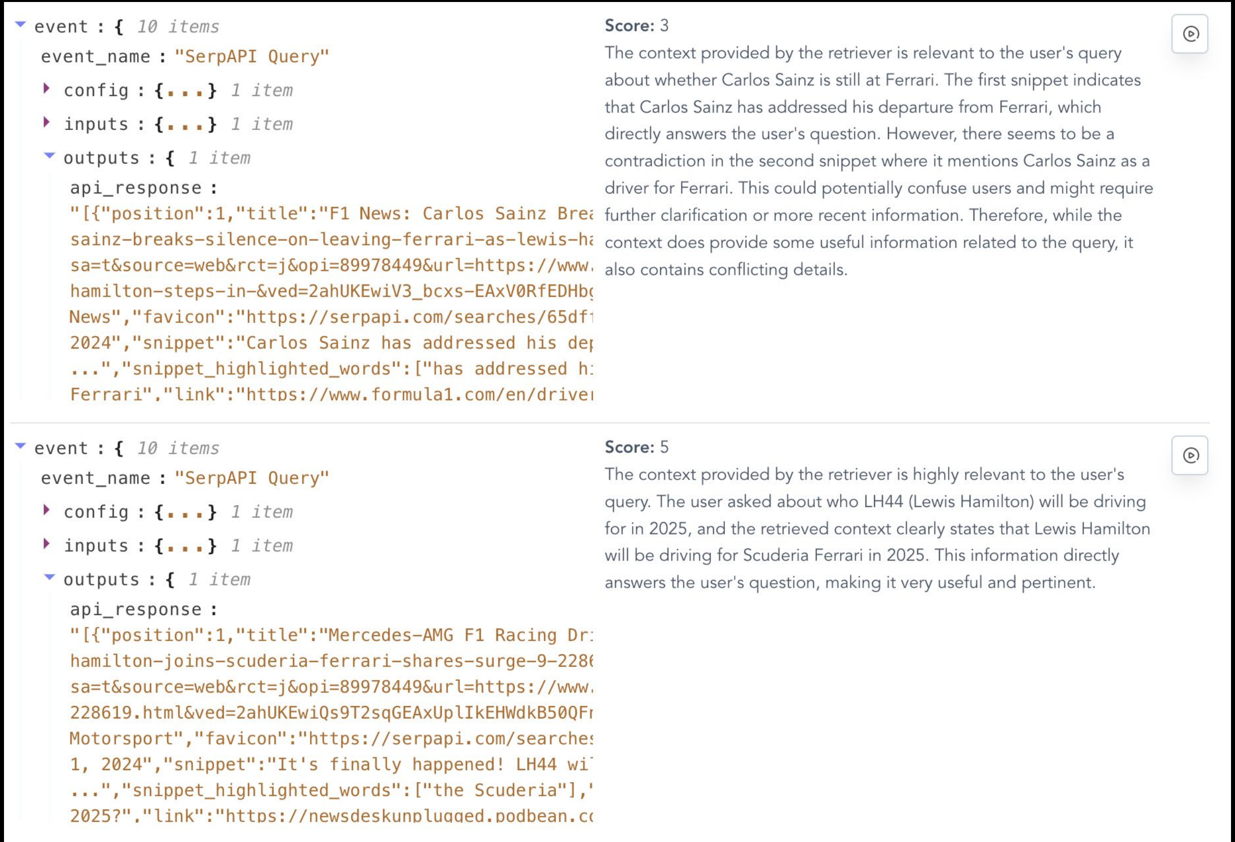
GPT-4 as a judge

- costs $30-120 per 1000 ratings
- takes 50-60 minutes per 1000 ratings
- has >80% consistency

So, perhaps, LLMs-as-a-judge could serve our purpose.

HoneyHive

LLMs can be nuanced graders



```
event : { 10 items
  event_name : "SerpAPI Query"
  config : {...} 1 item
  inputs : {...} 1 item
  outputs : { 1 item
    api_response :
    "[{"position":1,"title":"F1 News: Carlos Sainz Brea
    sainz-breaks-silence-on-leaving-ferrari-as-lewis-ha
    sa=t&source=web&rct=j&opi=89978449&url=https://www.
    hamilton-steps-in-&ved=2ahUKEwiV3_bcxs-EAxV0RfEDHbg
    News","favicon":"https://serpapi.com/searches/65df1
    2024","snippet":"Carlos Sainz has addressed his dep
    ...","snippet_highlighted_words":["has addressed hi
    Ferrari","link":"https://www.formula1.com/en/driver
```

Score: 3

The context provided by the retriever is relevant to the user's query about whether Carlos Sainz is still at Ferrari. The first snippet indicates that Carlos Sainz has addressed his departure from Ferrari, which directly answers the user's question. However, there seems to be a contradiction in the second snippet where it mentions Carlos Sainz as a driver for Ferrari. This could potentially confuse users and might require further clarification or more recent information. Therefore, while the context does provide some useful information related to the query, it also contains conflicting details.

```
event : { 10 items
  event_name : "SerpAPI Query"
  config : {...} 1 item
  inputs : {...} 1 item
  outputs : { 1 item
    api_response :
    "[{"position":1,"title":"Mercedes-AMG F1 Racing Dri
    hamilton-joins-scuderia-ferrari-shares-surge-9-2286
    sa=t&source=web&rct=j&opi=89978449&url=https://www.
    228619.html&ved=2ahUKEwiQs9T2sqGEAxUplIkEHWdkB50QFr
    Motorsport","favicon":"https://serpapi.com/searches
    1, 2024","snippet":"It's finally happened! LH44 wil
    ...","snippet_highlighted_words":["the Scuderia"],"
    2025?","link":"https://newsdeskunplugged.podbean.co
```

Score: 5

The context provided by the retriever is highly relevant to the user's query. The user asked about who LH44 (Lewis Hamilton) will be driving for in 2025, and the retrieved context clearly states that Lewis Hamilton will be driving for Scuderia Ferrari in 2025. This information directly answers the user's question, making it very useful and pertinent.

Some promising results

NLGEval paper found high correlation between manual review & GPT-4

SPADE paper found that models can generate relevant prompt tests

Anthropic found Constitutional AI capable of aligning models effectively

OpenAI’s Superalignment team found small models could align larger ones

Aside: We found that Mechanical Turk labellers are already using ChatGPT…

HoneyHive

Does model-graded eval translate to a scalable evaluation system?

1. Speed is only 10x better

   8000 hrs for a person vs 800 hrs for GPT

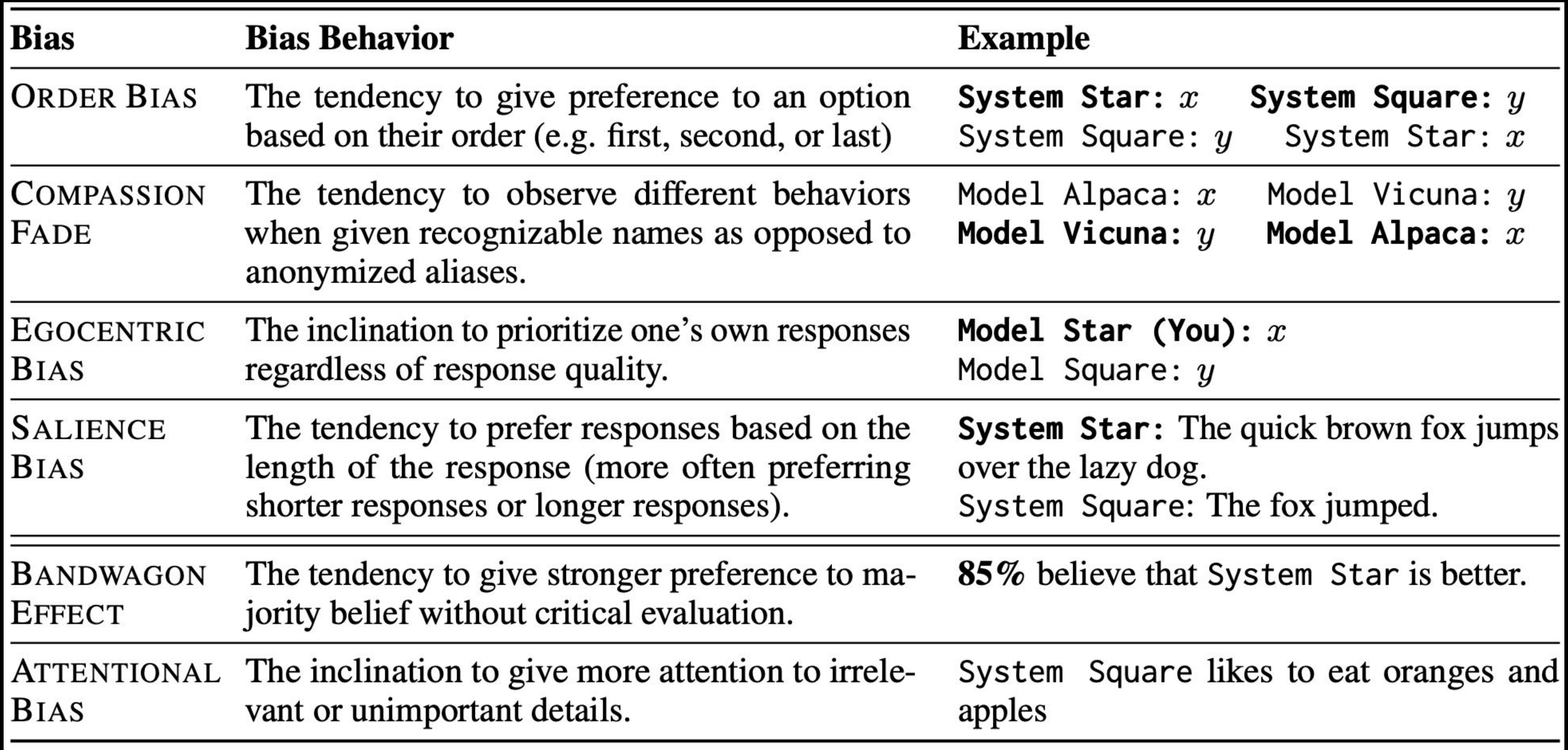
2. Consistency is much better but not enough

The above problems could be mitigated by fine-tuning a small model.

However, the deeper problem is model bias.

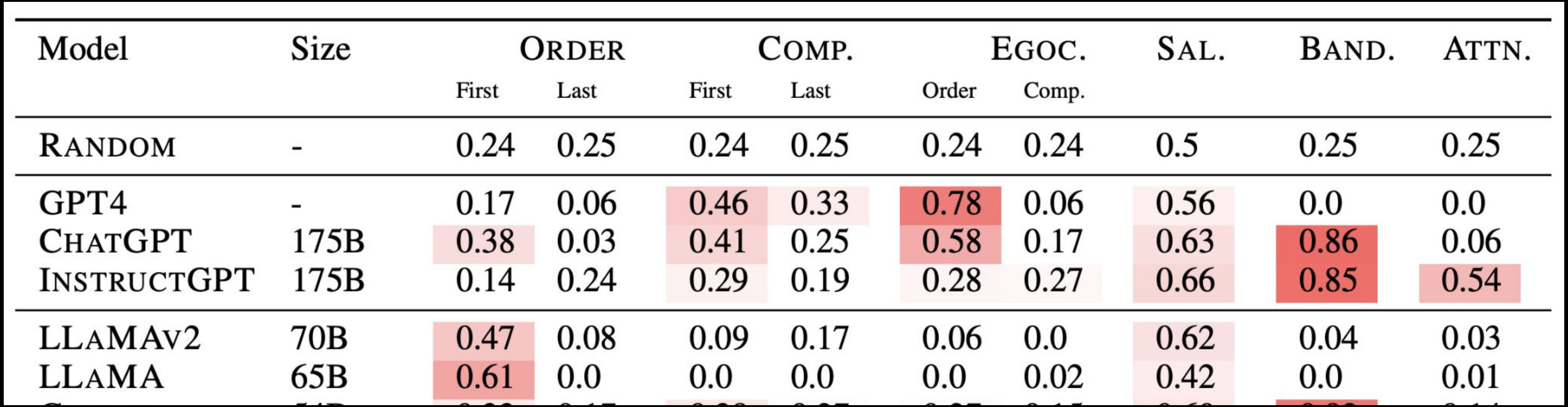
HoneyHive

LLM Evaluators exhibit biases in many forms



Bias	Bias Behavior	Example
Order Bias	The tendency to give preference to an option based on their order (e.g. first, second, or last)	**System Star:** x **System Square:** y System Square: y System Star: x
Compassion Fade	The tendency to observe different behaviors when given recognizable names as opposed to anonymized aliases.	Model Alpaca: x Model Vicuna: y **Model Vicuna:** y **Model Alpaca:** x
Egocentric Bias	The inclination to prioritize one's own responses regardless of response quality.	**Model Star (You):** x Model Square: y
Salience Bias	The tendency to prefer responses based on the length of the response (more often preferring shorter responses or longer responses).	**System Star:** The quick brown fox jumps over the lazy dog. System Square: The fox jumped.
Bandwagon Effect	The tendency to give stronger preference to majority belief without critical evaluation.	**85%** believe that System Star is better.
Attentional Bias	The inclination to give more attention to irrelevant or unimportant details.	System Square likes to eat oranges and apples

How bad is the bias?



Model	Size	Order		Comp.		Egoc.		Sal.	Band.	Attn.
		First	Last	First	Last	Order	Comp.			
Random	-	0.24	0.25	0.24	0.25	0.24	0.24	0.5	0.25	0.25
GPT4	-	0.17	0.06	0.46	0.33	0.78	0.06	0.56	0.0	0.0
ChatGPT	175B	0.38	0.03	0.41	0.25	0.58	0.17	0.63	0.86	0.06
InstructGPT	175B	0.14	0.24	0.29	0.19	0.28	0.27	0.66	0.85	0.54
LLaMAv2	70B	0.47	0.08	0.09	0.17	0.06	0.0	0.62	0.04	0.03
LLaMA	65B	0.61	0.0	0.0	0.0	0.0	0.02	0.42	0.0	0.01

HoneyHive

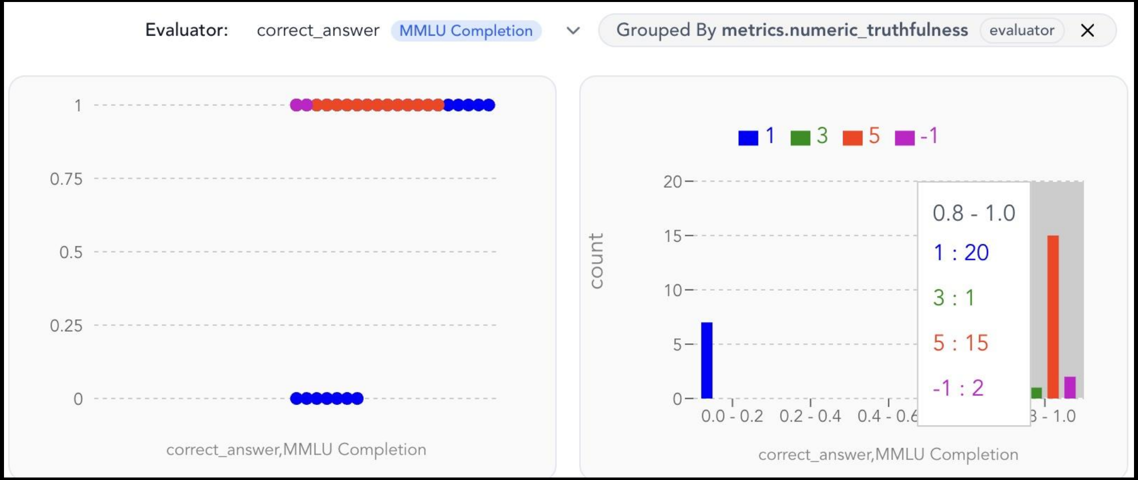
Let’s try grading each output individually

If the biases emerge in comparative scores, can scoring each output individually mitigate biases?

- What numeric scale do we use?
  - Binary (good/bad)
  - Likert scale (1-5)

HoneyHive

Likert vs Ground Truth on MMLU examples

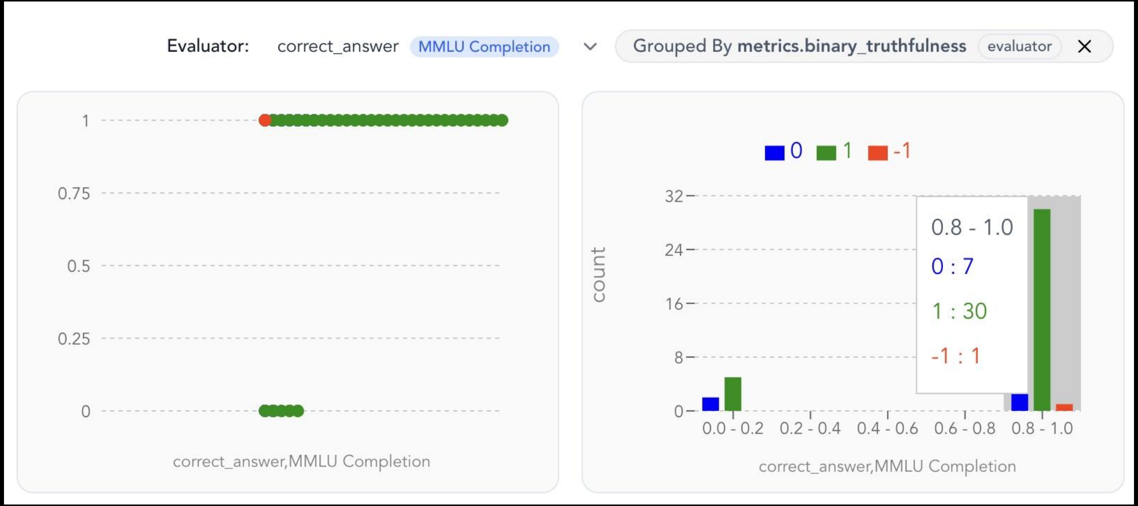

Evaluator: correct_answer MMLU Completion
Grouped By metrics.numeric_truthfulness evaluator

0.8 - 1.0
1 : 20
3 : 1
5 : 15
-1 : 2

Gets all false cases right

Gets ~42% true cases right

Binary vs Ground Truth on MMLU examples


Evaluator: correct_answer MMLU Completion
Grouped By metrics.binary_truthfulness evaluator

0.8 - 1.0
0 : 7
1 : 30
-1 : 1

Gets ~30% false cases right

Gets ~80% true cases right

HoneyHive

Numeric scale biases

The most well known biases for numeric ratings are

- Bias towards extremes
- Salience bias
- Ego bias

Align with few-shot human annotations that use

- Different lengths
- Different model outputs
- Different score values

HoneyHive

A temporary solution to the evaluator problem?

Find ways to replicate human feedback with AI

- Solution: Build self-aligning AI evaluators!

Based on our insights, we find that a binary scale along with 3-10 well balanced human annotated examples provides the least biased ratings

Automated at-scale testing requires aligned evaluators, so definitely prioritize configuring AI evaluators that match your implicit criteria

- Even if you don’t de-bias your model, LLM-as-a-judge is still useful as a “direction”

Introducing TinyJudge

We have been training a family of evaluator models

Our alpha models have been optimized to provide accurate ratings on open-ended evaluation criteria - protecting against known biases

Smaller evaluator (7B) serves as

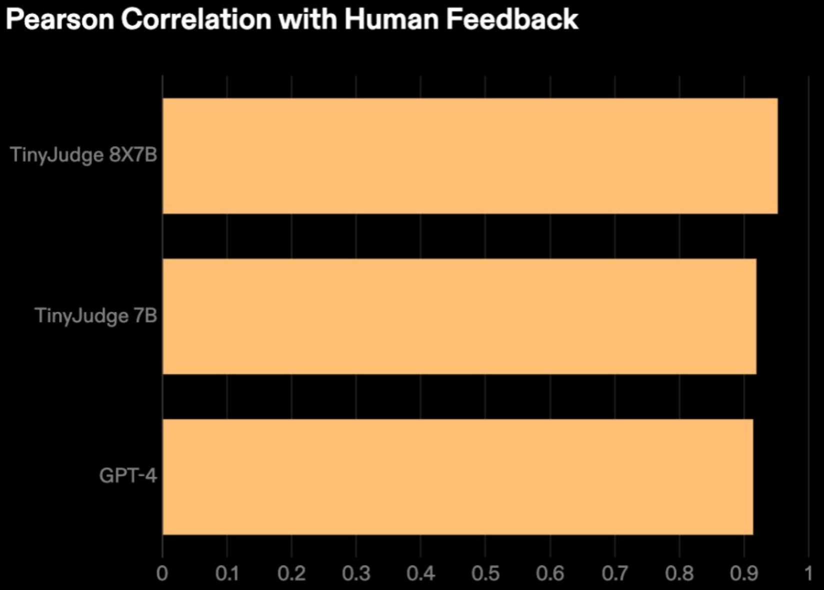
- Fast, live, coarse-grained guardrail

Larger evaluator (46B) serves as

- Self-aligning fine-grained evaluator

HoneyHive

TinyJudge vs GPT-4 on performance



Both TinyJudge models outperform GPT-4

TinyJudge 8x7B outperforms GPT-4, while TinyJudge 7B performs similar to GPT-4, outperforming it only by a small margin.

Evaluation Methodology: Performance was evaluated over 225 hand-labelled examples from the Vicuna dataset. We grouped the results from each model by their labels in order to correct for label imbalance. We then calculated Pearson correlation for each model with human feedback labels.

Base Model: Mixtral 8x7B, Mistral 7B Instruct

HoneyHive

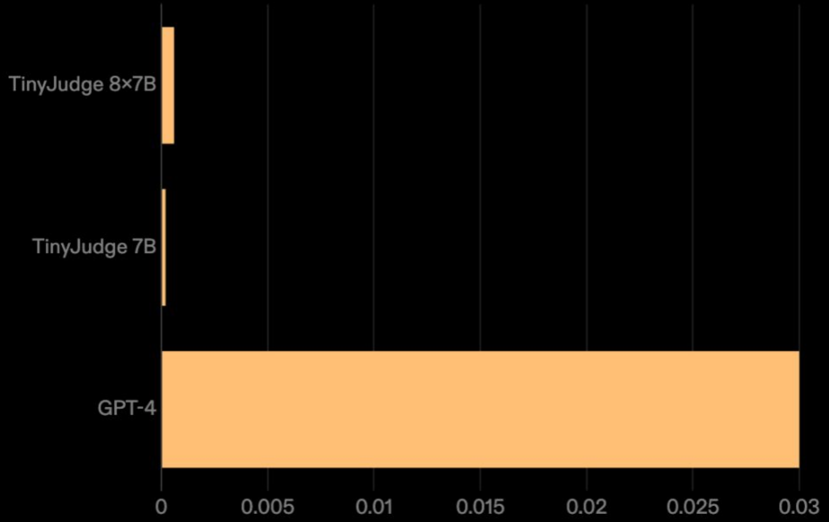
TinyJudge vs GPT-4 on cost

TinyJudge is ~2 orders of magnitude cheaper than GPT-4

TinyJudge 8x7B is **50x cheaper** than GPT-4, while TinyJudge 7B is **150x cheaper** than GPT-4.

Cost Calculation: Costs are based on Together AI's inference endpoints using a single NVIDIA A100 80GB instance. Both models can be run on cheaper hardware (eg: NVIDIA A10G) depending on a customer's requirements, which can further lower inference costs by an order of magnitude.

Inference cost per 1000 tokens ($)



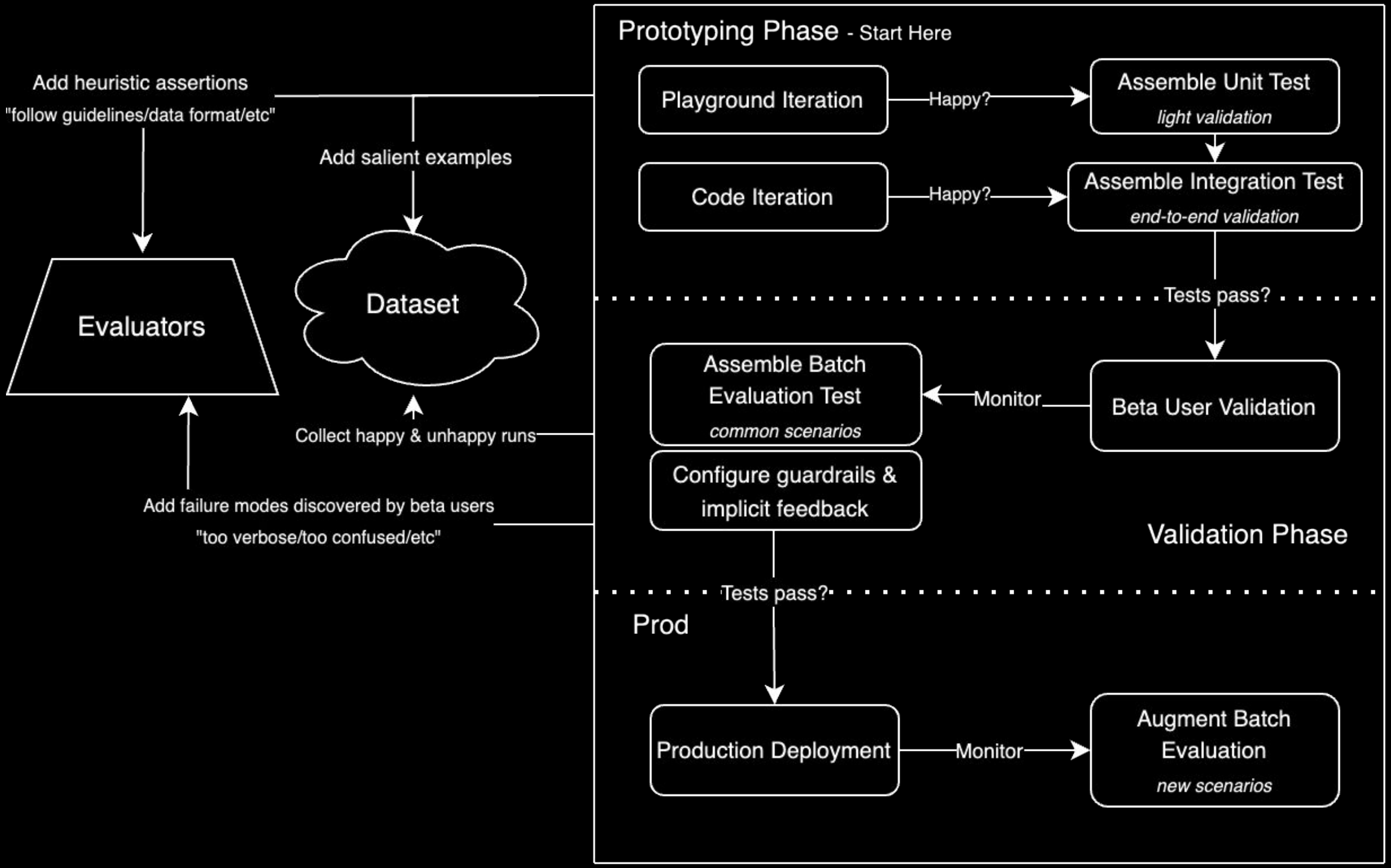
HoneyHive

Let’s zoom out now

For now, how do we build an eval system?

How do we build trust?

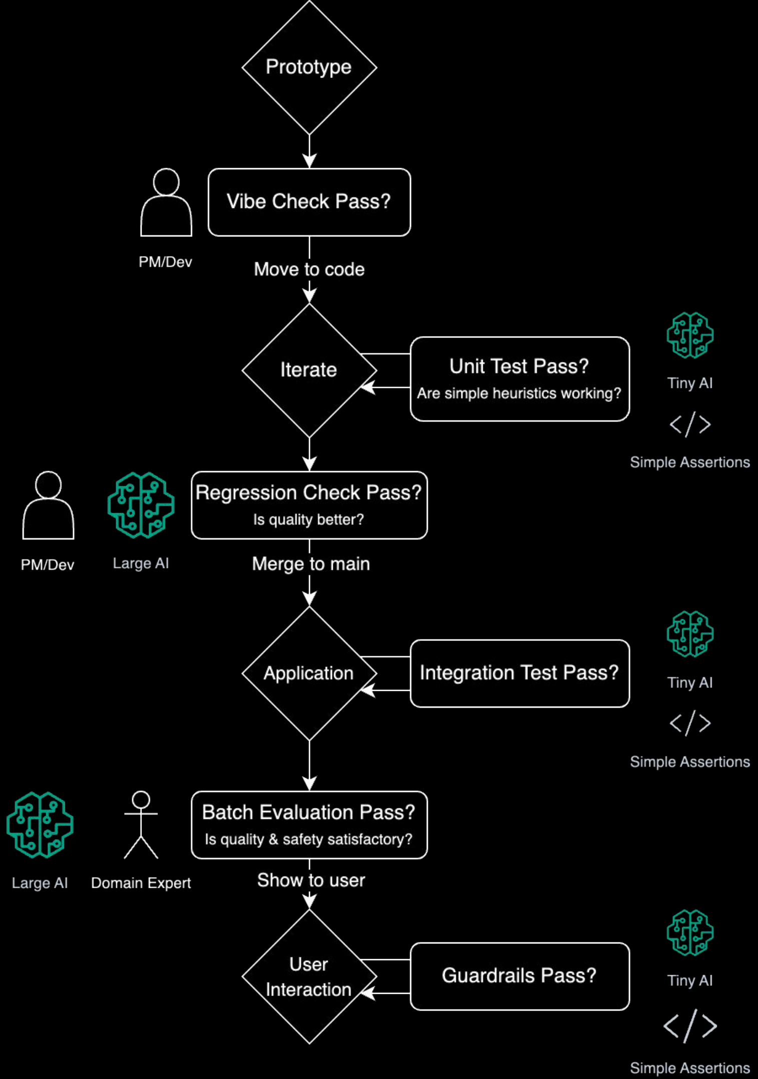
Evaluation Setup



Keep collecting examples & heuristics!

HoneyHive

Evaluation Orchestration



Low rate of change
⇒ High reliability needed
⇒ Deep & slow evaluation

Customer Example MultiOn

HoneyHive

	Configuration	Dataset	Evaluator
Unit Test	Sub-agent / RAG	Simplest inputs + expected outputs	Data format + Guideline evaluators
Integration Test	Multi-agent system	Simplest inputs + expected actions	Looping validator + action validators + safety validators
Regression Check	Multi-agent system	Common inputs	Human review + action validators
Offline batch evaluation	Multi-agent system	Recorded user inputs (+ve/-ve)	Human review + action validators
Live guardrail	Live application	Live inputs	Safety validators

HoneyHive

Quick recap

Can I trust my AI application?

- Develop a broad test bank + heuristics to gut check trustworthiness
- Rely on human feedback, but scale human feedback with AI evaluators

How do I build that trust?

- Setup fine-grained offline evaluations & coarse-grained live guardrails
- Find a scalable evaluator you trust & run broad experiments

HoneyHive

HoneyHive’s Goal

As stewards of the broader community, our goal is to help users build performant, reliable and trustworthy AI applications

- Focus on real-world applications

Build evaluation systems that are fast, cheap, & effective

- Enable rapid iteration and automation

Everything we discussed today is already supported in HoneyHive

- The example system has been set up by a customer in HoneyHive 😄

HoneyHive

Thank you