GROUPBRAIN TALKS · JULY 2026

LLM Application Evaluations

Measuring the product we build on top of a model — repeatably, systematically, and against written criteria.

Contents

01

The Problem With Vibe Testing

Why the default way of checking an LLM app quietly fails.

Two Problems With Vibe Testing

01

Not repeatable

Run the same prompt tomorrow — or hand it to a teammate — and you get a different judgement. There is no fixed set of inputs and no recorded expectation, so you cannot reproduce a result or prove that a change actually helped.

02

Subjective and informal

“Good” means something different to every reviewer, and nothing is written down. Without shared criteria the verdict depends on who happened to look and what mood they were in — not on the system itself.

False Confidence

A prompt tweak improves the one case you happened to check.

It quietly degrades ten edge cases nobody looked at.

There is no baseline, and no record of what “working” looked like before.

Why LLM Apps Are Harder to Evaluate

DIMENSION	TRADITIONAL ML / DL	LLM APPLICATIONS
Outputs	Deterministic-ish. A prediction to compare against a label.	Probabilistic and open-ended. The same prompt yields many valid — and invalid — free-text answers, so there is no single correct string to match.
Benchmark	A single scalar — accuracy, F1, AUC — captures “how good”.	Multi-dimensional. Correctness, faithfulness, tone, safety, latency and cost each need their own setup; no one number sums them.

What an Evaluation Is

Repeatable
The same inputs and the same procedure, every single run — so two results are comparable.

Systematic
Applied the same way to every candidate, not only to the outputs that caught someone's eye.

Written down
Clear criteria for what "good" means, agreed before the scoring starts.

An eval is not just a metric — it is the entire testing setup built around a question.

The Testing Setup, in Four Parts

1

A question

What exactly are we measuring?

2

A dataset

The inputs we test against.

3

A method

How each output is scored.

4

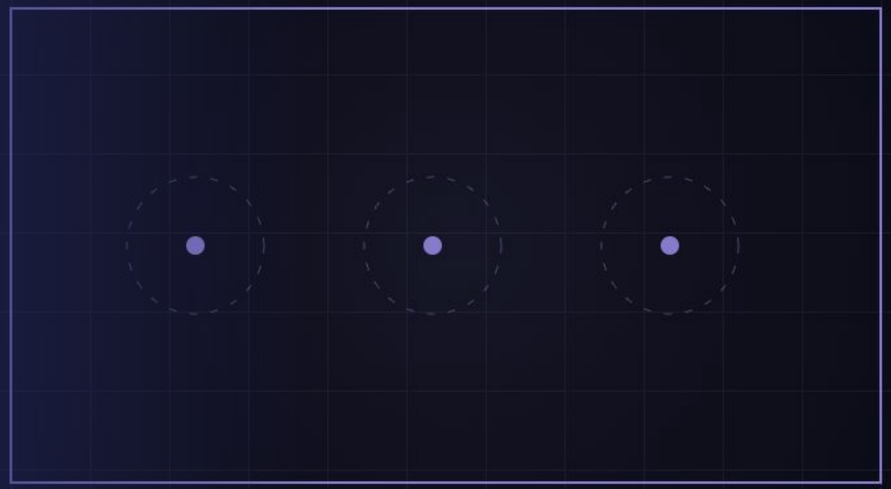
A verdict

Metrics we can act on.

02

Scope and Workflow

Two families of eval, three levels of scope, one seven-step loop.



Model Evals vs. Application Evals

Model evaluations

Measure the raw model — reasoning, knowledge, safety — on broad academic benchmarks such as MMLU, GSM8K and SWE-bench. This is what model providers publish.

OUR FOCUS

Application evaluations

Measure the product built on top of a model — the prompts, retrieval, tools and glue — against the job it must actually do for real users.

Three Levels of Application Eval

Application level

The whole product in production — latency, cost, safety, user outcomes.

Whole-system / workflow level

Components working together, end to end.

- **Component** — e.g. retriever
- **Component** — e.g. generator
- **Component** — e.g. tool call

The Seven-Step Eval Loop

01

Decide what you evaluate

Pin down the exact component, workflow or application-level question.

02

Define success criteria

Write down what “good” means and how it will be scored.

03

Create the golden dataset

Curate trusted inputs, and expected outputs where relevant.

04

Choose a method

LLM-as-a-judge, programmatic, human eval, or a mix.

05

Run the system

Execute over the dataset and collect every output.

06

Score the results

Apply the scoring method to produce metrics.

07

Analyse, ship, monitor

Act on findings, deploy, then keep evaluating in production.

↺

Back to step one

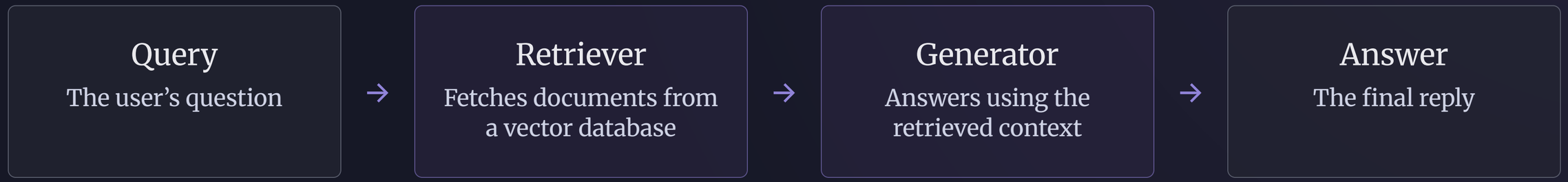
Production surprises become the next question.

03

One Application, Several Evals

A RAG chatbot, evaluated three times over — and why each pass is not enough.

A RAG Chatbot, End to End



The two lit boxes are the obvious failure points — and each earns its own eval pipeline.

One Eval per Component

COMPONENT	HOW IT FAILS	THE QUESTION THE EVAL ASKS	METRIC
Retriever	Fetches the wrong or irrelevant documents.	Given a query, are the relevant documents actually being retrieved?	Context Relevance
Generator	Ignores correct context and hallucinates, or answers incorrectly.	Given a context, is the answer grounded in it, without inventing outside facts?	Faithfulness

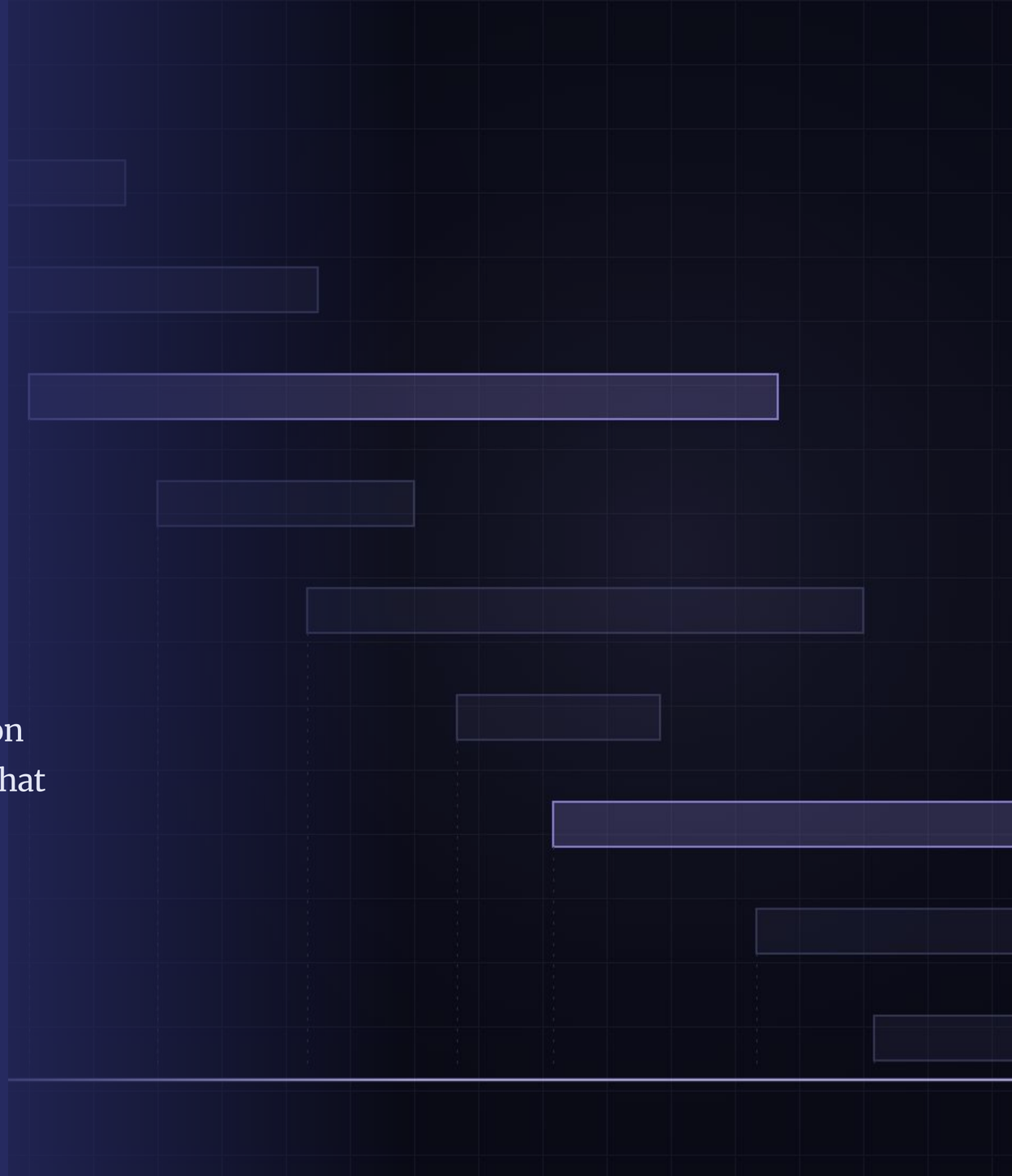
Where Component Evals Stop

WITHOUT A RE-RANKER		WITH A RE-RANKER
1 · D1		1 · D5 — promoted to top
2 · D2		2 · D1
3 · D3	→	3 · D2
4 · D4		4 · D3
5 · D5 — truly relevant		5 · D4

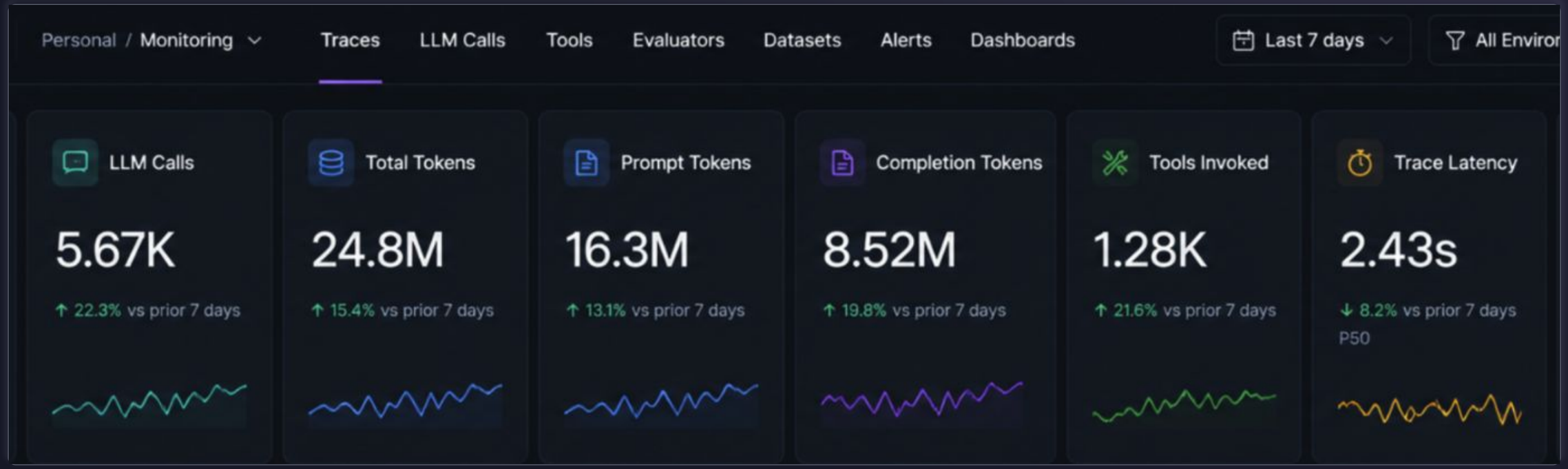
Retriever and generator each pass in isolation; the combination fails. Only a workflow-level eval sees it.

10s

End-to-end latency for a single query on a correct retriever-plus-generator workflow — far too slow for a production chatbot. Correctness alone does not make a product usable, and that failure is only visible at the application level.



Tracing the Application Level using LangSmith

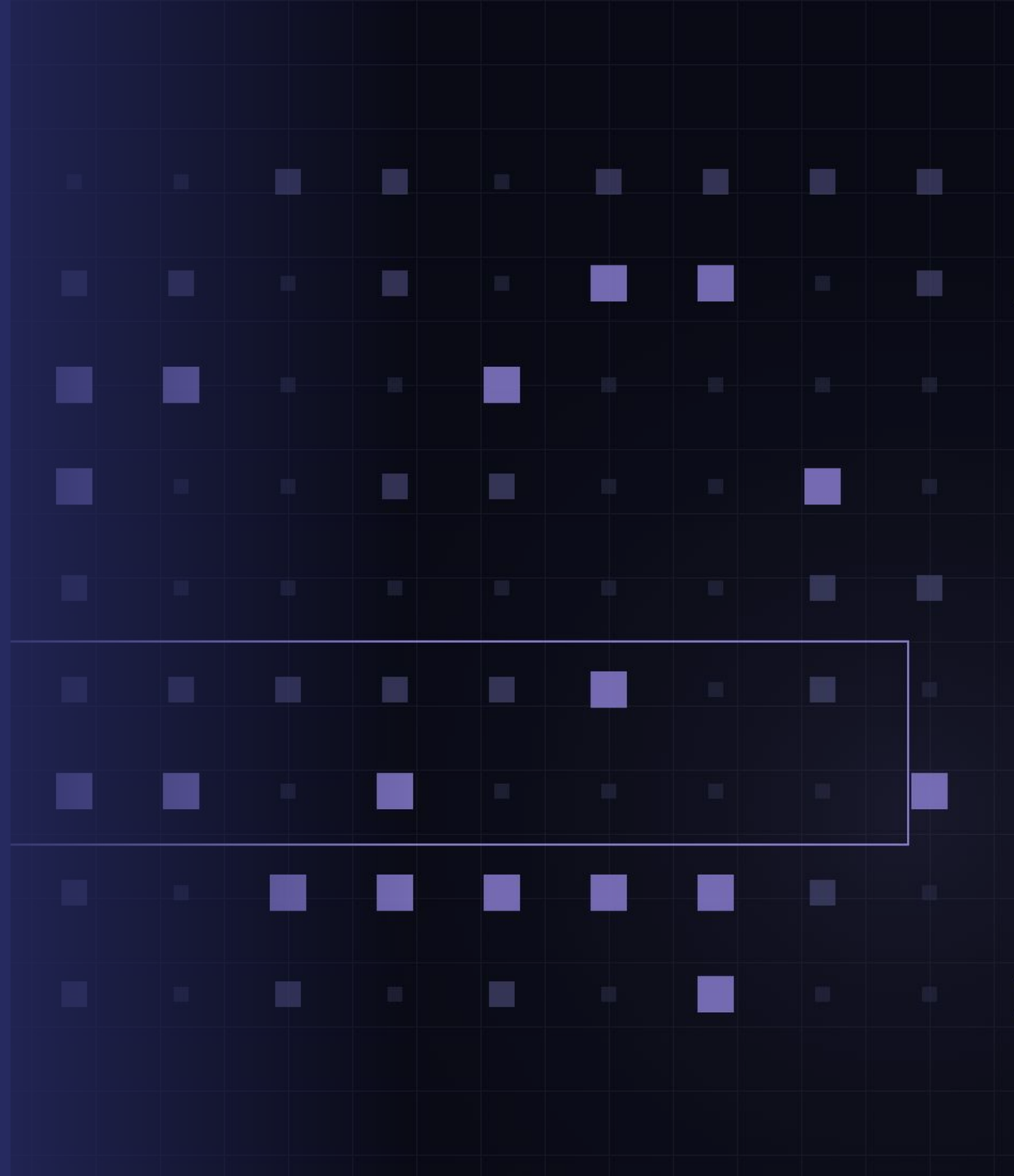


A tracing dashboard makes application-level metrics observable: LLM calls, token spend, tool invocations and P50 latency, each against the prior week.

04

Methods and Metrics

Three methods, two timings, and the metrics that keep judges honest.



Three Eval Methods

01

LLM-as-a-judge

A capable model scores the output against a rubric — scalable and flexible for open-ended, subjective qualities.

02

Programmatic

Deterministic code metrics — Recall@k, exact match, regex, JSON-schema checks. Cheap, fast, perfectly repeatable.

03

Human eval

People judge quality directly — the gold standard, but slow and costly, so it is spent deliberately.

Choosing a Method

Is the answer checkable in code?	YES →	**Programmatic** Recall@k, exact match, schema checks
Subjective, but high volume?	YES →	**LLM-as-a-judge** Rubric scoring, validated against humans
High-stakes, or need ground truth?	YES →	**Human eval** Golden sets, red teaming, A/B, human-in-the-loop

“Human evaluation remains the ground truth you calibrate everything else against.”

— red teaming, A/B testing and human-in-the-loop all sit under it

Offline vs. Online Evaluation

Offline evaluation

BENEFITS

Fully repeatable — the same fixed dataset every run
Fast to iterate and cheap to re-run in CI

TRADE-OFFS

Misses real-world inputs you never anticipated
Good offline scores do not guarantee live success

Online evaluation

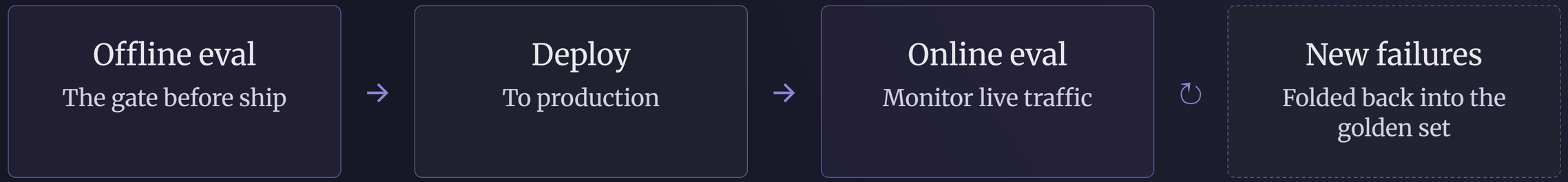
BENEFITS

Reflects true production behaviour and real users
Catches drift and edge cases as they emerge
Enables A/B tests and live guardrail monitoring

TRADE-OFFS

Not repeatable — traffic is never identical twice
Risky — real users may hit a bad response

The Offline–Online Loop



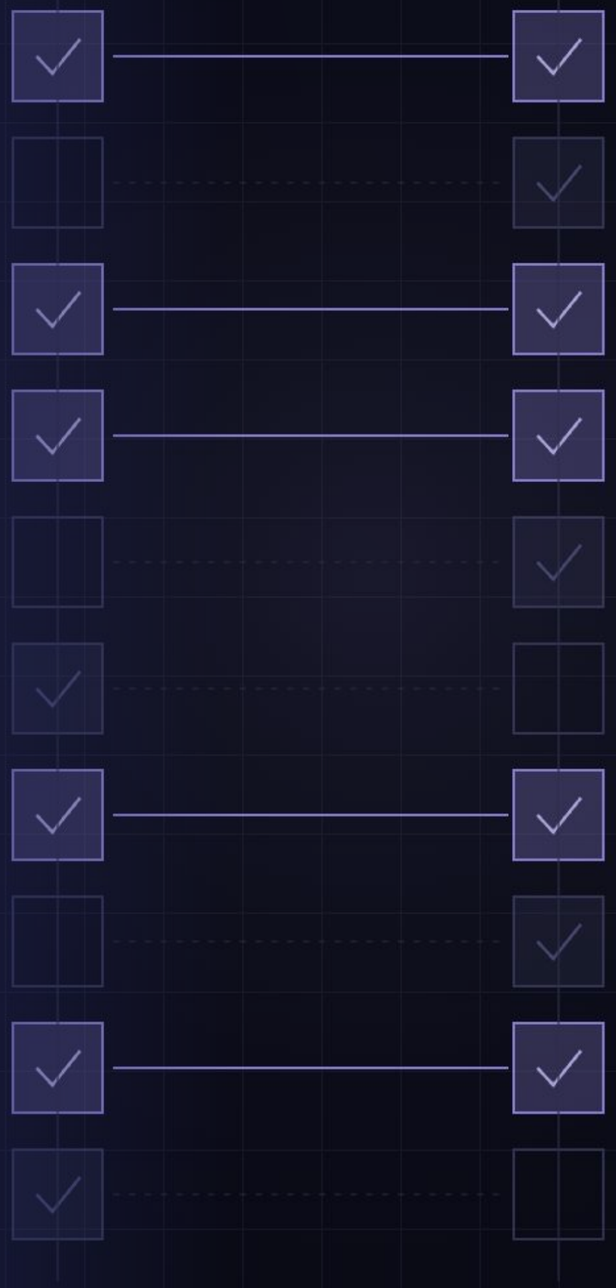
Offline evals gate the release, online evals watch what really happens, and every surprise makes the gate stronger.

Inter-rater Agreement Metrics

METRIC	WHAT IT MEASURES	WHEN TO REACH FOR IT
Agreement rate	The simple proportion of the time raters give the same response.	A first look — quick, but flattering.
Cohen’s Kappa	Adjusts the observed agreement rate for what would happen by random chance.	Two raters on a categorical label.
Fleiss’ Kappa · Krippendorff’s Alpha	Extensions of Kappa for more than two raters.	Panels, and partially-overlapping rating sets.

50%

Agreement two coin-flippers reach by pure chance on a binary label. On tasks with few options raters agree a lot for no reason at all — Cohen’s Kappa subtracts that expected-by-chance agreement, so a high Kappa means they genuinely share the same standard.



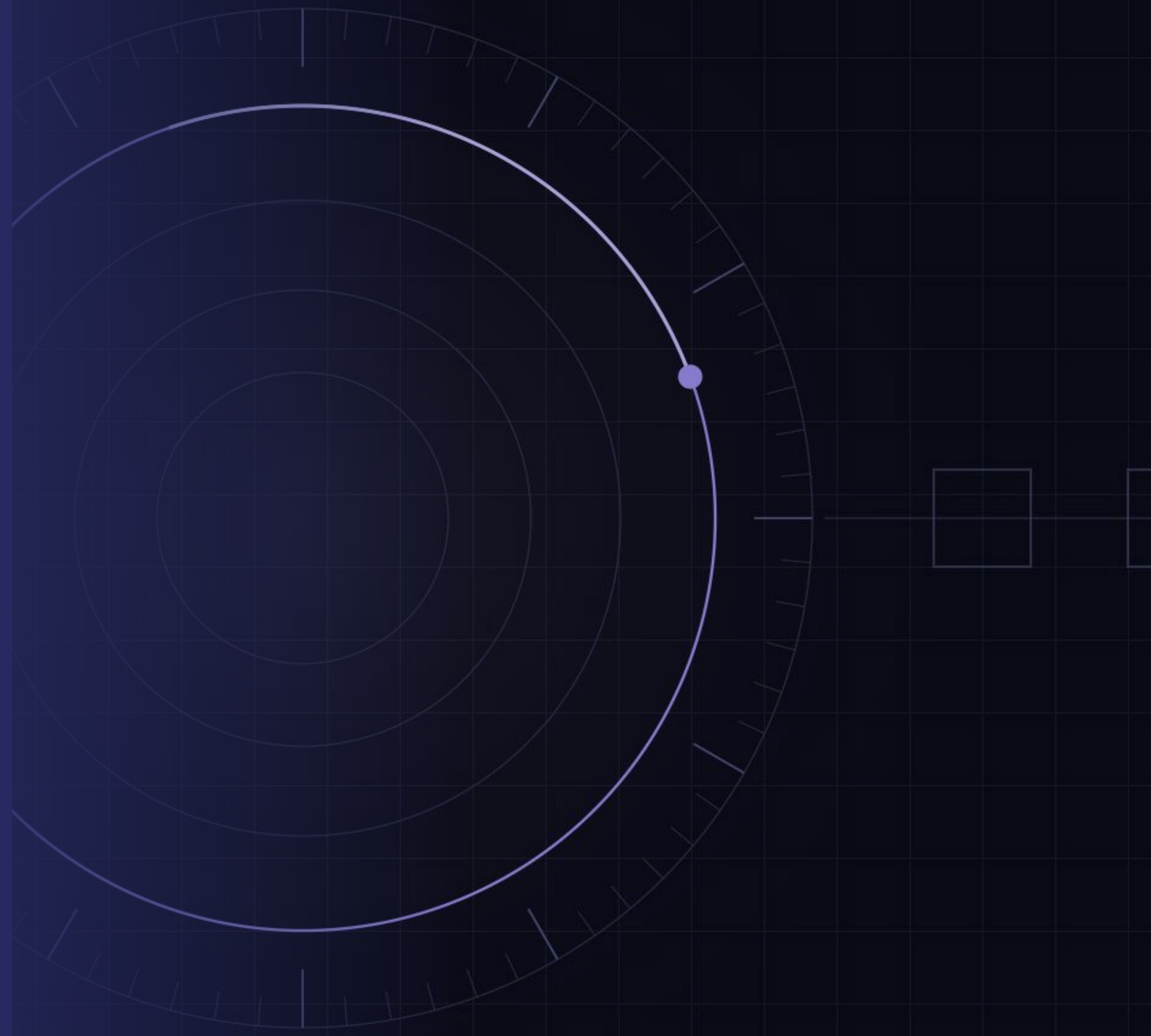
Rule-based Metrics

METRIC	STANDS FOR	WHAT IT DOES
METEOR	Metric for Evaluation of Translation with Explicit Ordering	Combines precision, recall and a penalty for poor word ordering. -> A better version of BLEU
BLEU	Bilingual Evaluation Understudy	Precision-focused n-gram comparison against the reference, with a brevity penalty.
ROUGE	Recall-Oriented Understudy for Gisting Evaluation	Measures overlap with the reference text; the usual choice for summarisation.

05

In Practice — Project Sakhi

Each level of the stack, the failure it hides, and the eval that catches it.



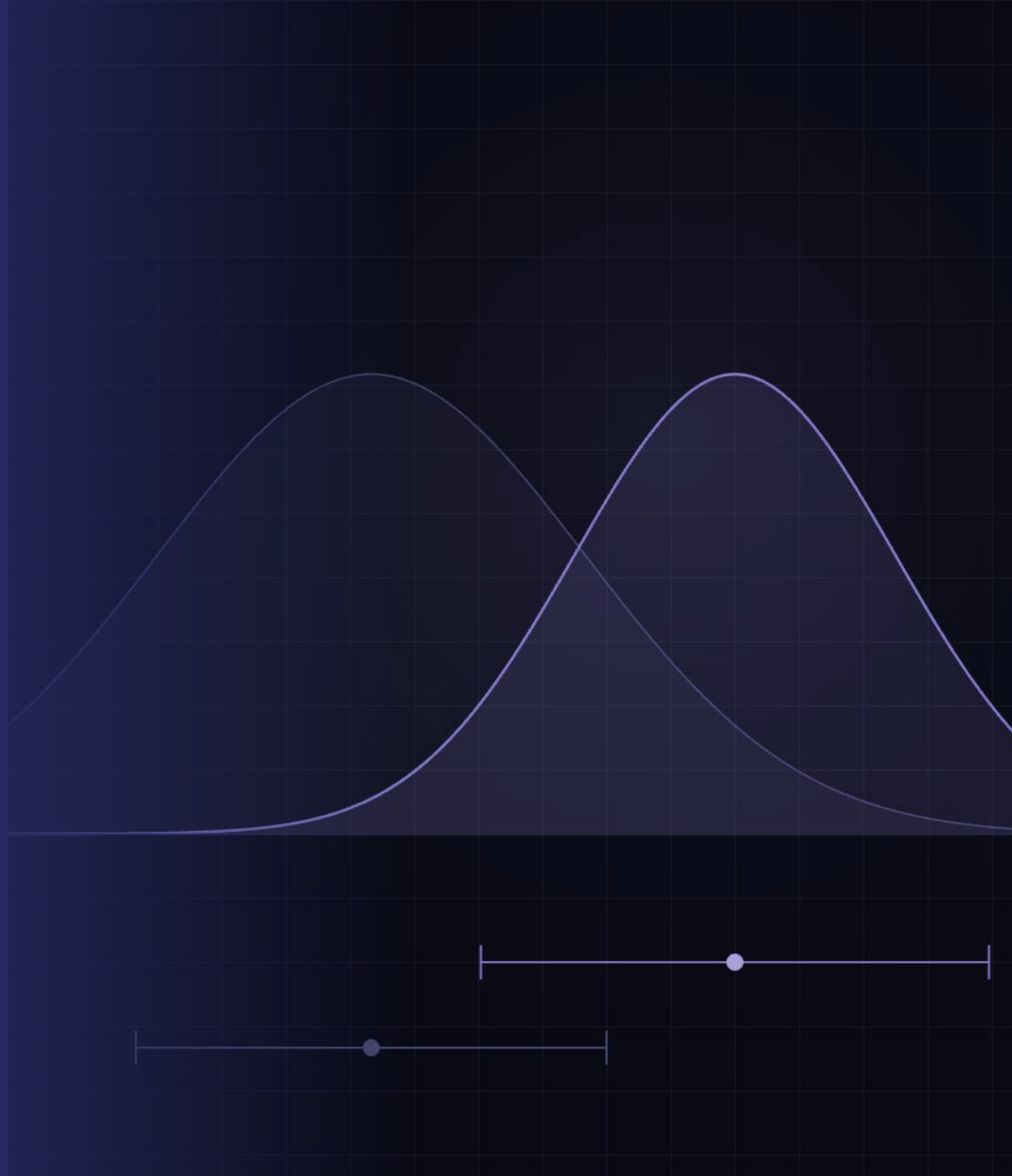
Sakhi, Evaluated at Every Level

COMPONENT	FAILURE IT CAN HIDE	EVAL THAT CATCHES IT
RAG retriever	Pulls the wrong lab-manual section, so guidance drifts from the syllabus.	Context Relevance · Recall@k
Tutoring LLM	Ignores the manual and invents a plausible-sounding but wrong step.	Faithfulness · Groundedness
Full guided session	Components each pass, yet the end-to-end explanation confuses a beginner.	Workflow-level eval on a golden set
Voice and memory loop	Correct but too slow, or recalls the wrong student's history.	Application-level latency and recall %

06

Research Papers

RAGAS and ARES — automating RAG evaluation without human labels.



RAGAS — Reference-Free RAG Evaluation

Faithfulness

The LLM extracts atomic statements from question and answer, then checks each one against the retrieved context.

F = verified ÷ total statements

Answer Relevance

The LLM reverse-engineers n questions from the answer; each is embedded and compared with the original query.

AR = mean cosine similarity

Context Relevance

The LLM extracts the sentences of the context that were crucial to answering, and counts them.

CR = extracted ÷ total sentences

Es, James, Espinosa-Anke & Schockaert (2023) · arXiv:2309.15217 · open-source *ragas* Python library

ARES — Automated Evaluation for RAG

STAGE 01

Synthetic dataset

An in-domain passage set and a few-shot Q&A prompt drive a generator; queries are filtered by a retrieval check into positive and negative training triples.

STAGE 02

LLM judges

A lightweight model is contrastively fine-tuned into three judges — context relevance, answer faithfulness and answer relevance.

STAGE 03

Ranking with PPI

Judges score sampled triples; prediction-powered inference corrects them against a small human validation set, yielding ranked systems with confidence intervals.

Saad-Falcon, Khattab, Potts & Zaharia (Stanford) · NAACL 2024 · arXiv:2311.09476 · code at stanford-futuredata/ARES