



# **Building Trust in LLM Products: A Production-Ready Evaluation Framework**

LREC2026 – Industry day

# Speaker s



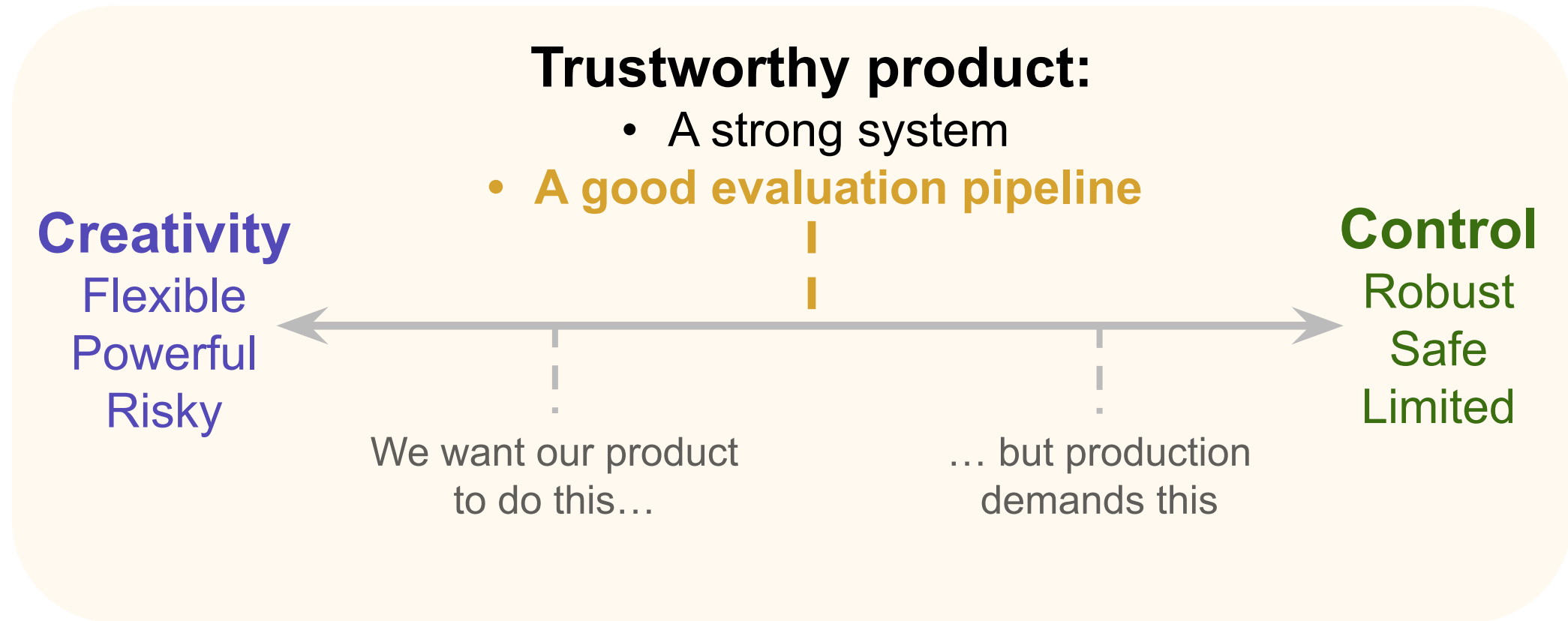
**Erin Pacquetet**  
**PhD**  
**Senior AI**  
**Scientist**



**Christophe**  
**Séjourné**  
**Head of AI**  
**& Data**

# Evaluation is important

- It's easy to do an AI POC... and it's difficult to ship to production



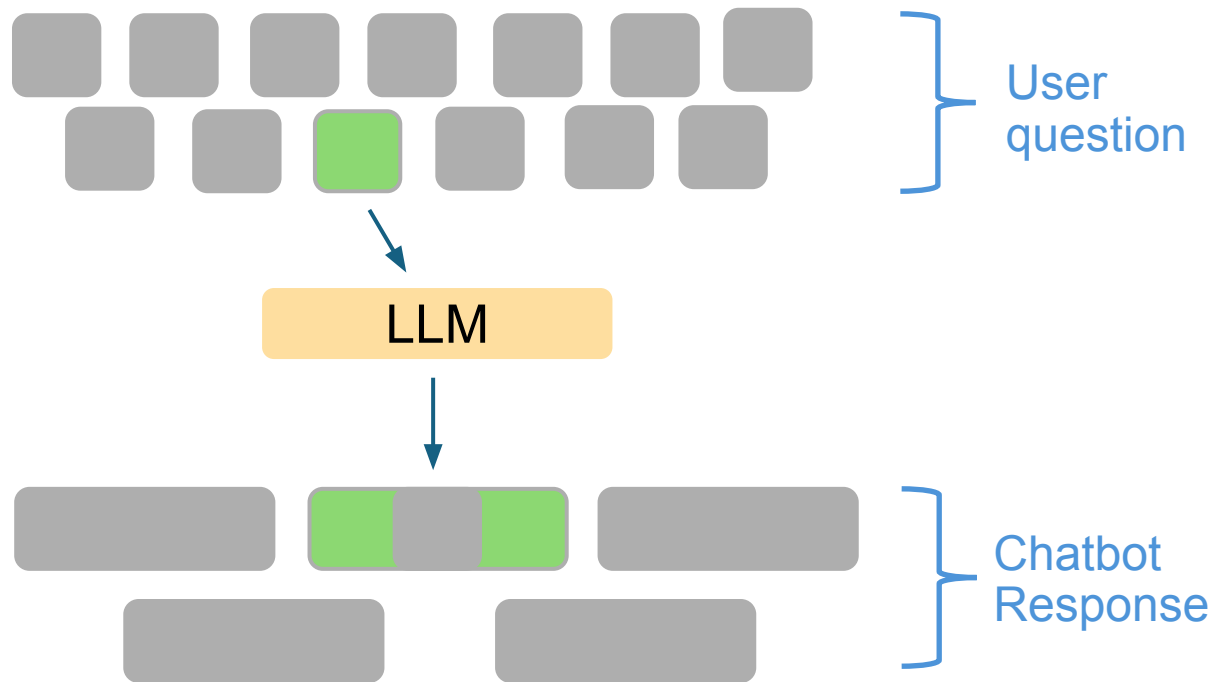
How do we trust the system?

**"Are we sure that it works well?"**

What is "well"?

# LLMs in products

- The difficulty in evaluating products with LLM comes from the very reason we use them.



LLMs are used when we have:

- A wide variety of **inputs**:

- That we don't control
- That the system has never seen

Potentially of poor quality

- **Outputs tailored to inputs**

# LLMs in products

- The difficulty in evaluating products with LLM comes from the very reason we use them.

We don't have control over the inputs.

We don't have control over the outputs.

LLMs are used when we have:

- A wide variety of **inputs**:

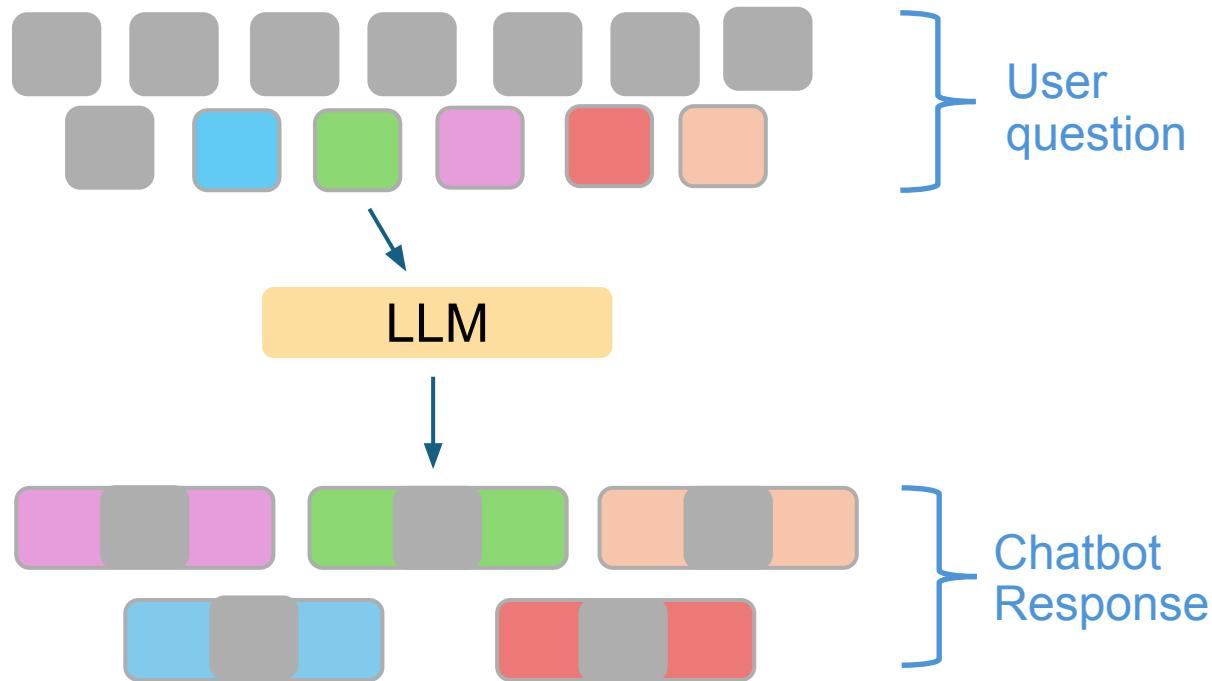
- That we don't control
- That the system has never seen

Potentially of poor quality

- **Outputs tailored to inputs**

# Evaluation pipeline

- Transform a manual qualitative assessment into a **quantitative and comprehensive measure** that can be run whenever you want, independently.



Evaluation :

1. *Simulate inputs*
2. *Generate outputs*
3. *Evaluate outputs*

When:

- *Before shipping to prod*
- *During the development phase*

# Why evaluation is hard

①

**Identify usage scenarios**

②

**Judge product behavior**

# Why evaluation is hard

①

**Identify usage scenarios**

**Identifying usage scenarios is hard because...**

- LLMs accept flexible, uncontrolled inputs; the input space is **effectively infinite**

→ Address by ***decomposing the system*** into testable components with well-scoped input boundaries

# Why evaluation is hard

②

**Judge product behavior**

**Judging behavior is hard because...**

- LLMs produce open-ended, generative outputs; correctness is **rarely binary**
- **Hallucinations** can occur and are hard to detect

→ *Address by combining heuristics, model-based graders, and human review tiered by stakes and severity*

# What is in our poster :

## WHY

Trust is a **production prerequisite**: PoCs are easy, production is hard

## WHAT

**Decompose into components** with bounded input spaces, then cover ideal, realistic, and adversarial usage scenarios

## HOW

**Score automatically**, calibrated against human judgment : scalable & repeatable

## ROADMAP

A **practical pipeline** built jointly by product & dev teams, never complete, always improving

### Building Trust in LLM Products A Production-Ready Evaluation Framework

Erin Pacquetet, Christophe Séjourné - SCIAM, Paris (France)  
erin.pacquetet@sciam.fr, christophe.sejourné@sciam.fr



#### WHY

Why do we need to evaluate GenAI products?

Trust as a deployment prerequisite

Why evaluation matters

PoCs are easy, production is hard. Trust is a prerequisite for deploying a product in production. A strong evaluation pipeline helps build that trust. Evaluation must fulfill two aspects :



Our framework helps teams build, measure and manage that trust.

The LLM double-edged sword

Why LLM products need a specific approach

LLMs are flexible and creative, making them ideal for powering products. But those same strengths make ① and ② harder to achieve.



① Identifying usage scenarios is hard because...

• LLMs accept flexible, uncontrolled inputs; the input space is effectively infinite

→ Address by decomposing the system into testable components with well-scoped input boundaries

② Judging behavior is hard because...

• LLMs produce open-ended, generative outputs; correctness is rarely binary  
• Hallucinations can occur and are hard to detect

→ Address by combining heuristics, model-based graders, and human review tiered by stakes and severity

Evaluating the product as a whole

Good models don't guarantee good products

Without operational quality the product is unusable; without output quality it's unreliable.

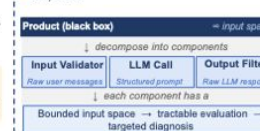


#### WHAT

What structure evaluation must follow

Decomposing the system

Addressing ① : bounding the scenario space per component



Decomposition approximate an infinite evaluation problem into a finite one. One component at a time.

3 families of scenarios to cover

Identify usage scenarios for each component

✓ Ideal — "Perfect" input  
Verifies core functionality under best-case conditions

✓ Realistic — Expected production input  
Tests robustness to noise in real-world settings

✓ Adverse — Aimed at breaking the system  
Probes security, safety and guardrail gaps

Each family has its own acceptance threshold. These scenarios form the Evaluation Dataset: a list of inputs and associated expected outputs, one set per component.

The hierarchy of judgment

Addressing ② : from inaccessible truth to automatic scorer

• The Oracle  
Ideal evaluator — inaccessible by definition

approximated by

• Human Evaluation  
Gold standard — Business & domain experts

→ Named → Close to truth  
→ Slow & costly → Variance & bias

calibrated & scaled by

• Automatic Scorers  
Observable proxies of quality

→ Scalable → Repeatable  
→ Partial → Imperfect

Every score must be validated against human judgment. A business pass must be an evaluation pass.

#### HOW

How to implement and run the evaluation

From judgment to metrics

The anatomy of a scorer

Each scorer implements a rubric and produces a metric :

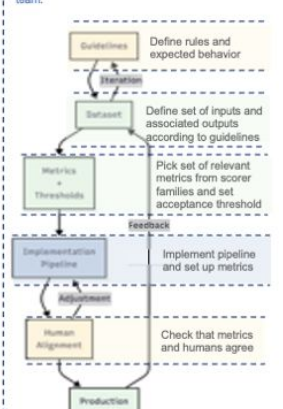
• Scorer families : Deterministic, Lexical, Semantic, LLM-based, Human  
• Each scorer family has its pro & con : precision, recall, scalability, cost, human alignment ability...  
• Scorers only evaluate one behavior of the component. Metrics must be combined to have a complete picture

| Rubric<br>What to verify               | Scorer<br>How to verify | Metric<br>Observable score |
|----------------------------------------|-------------------------|----------------------------|
| Is the answer faithful to the sources? | LLM-as-a-Judge          | True or false              |
| Is the relevant URL in the response?   | String matching         | True or false              |
| ...                                    | ...                     | ...                        |

Practical roadmap

Who needs to do what, in which order

The evaluation pipeline is built jointly by the business and product teams, and the developer team.



The evaluation pipeline is never complete, and that's the point :

• Every release adds new test cases  
• Edge cases today become guardrails tomorrow  
• The pipeline gets sharper as the product matures

