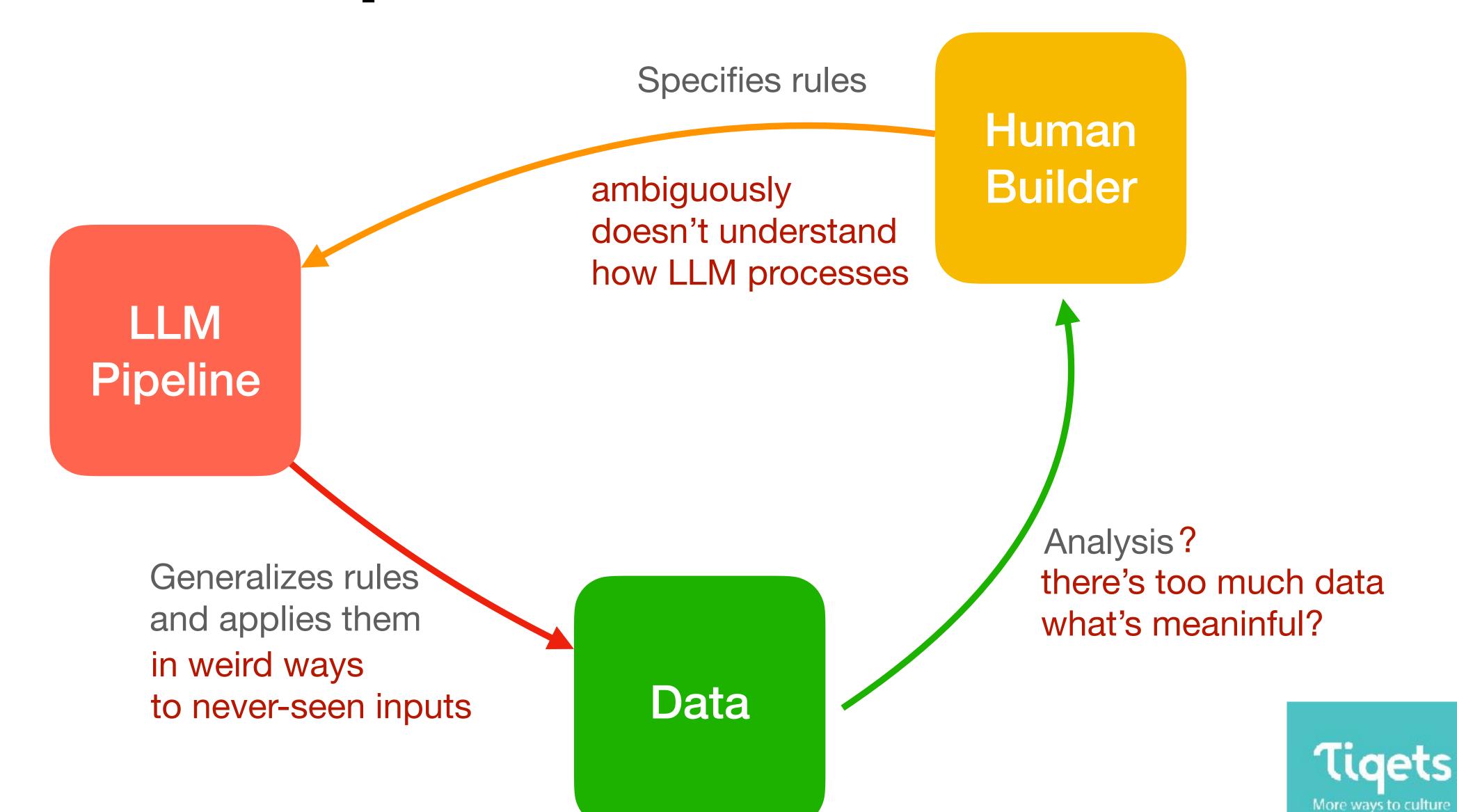
Evaluation-Driven Product Development

aiGrunn 2025

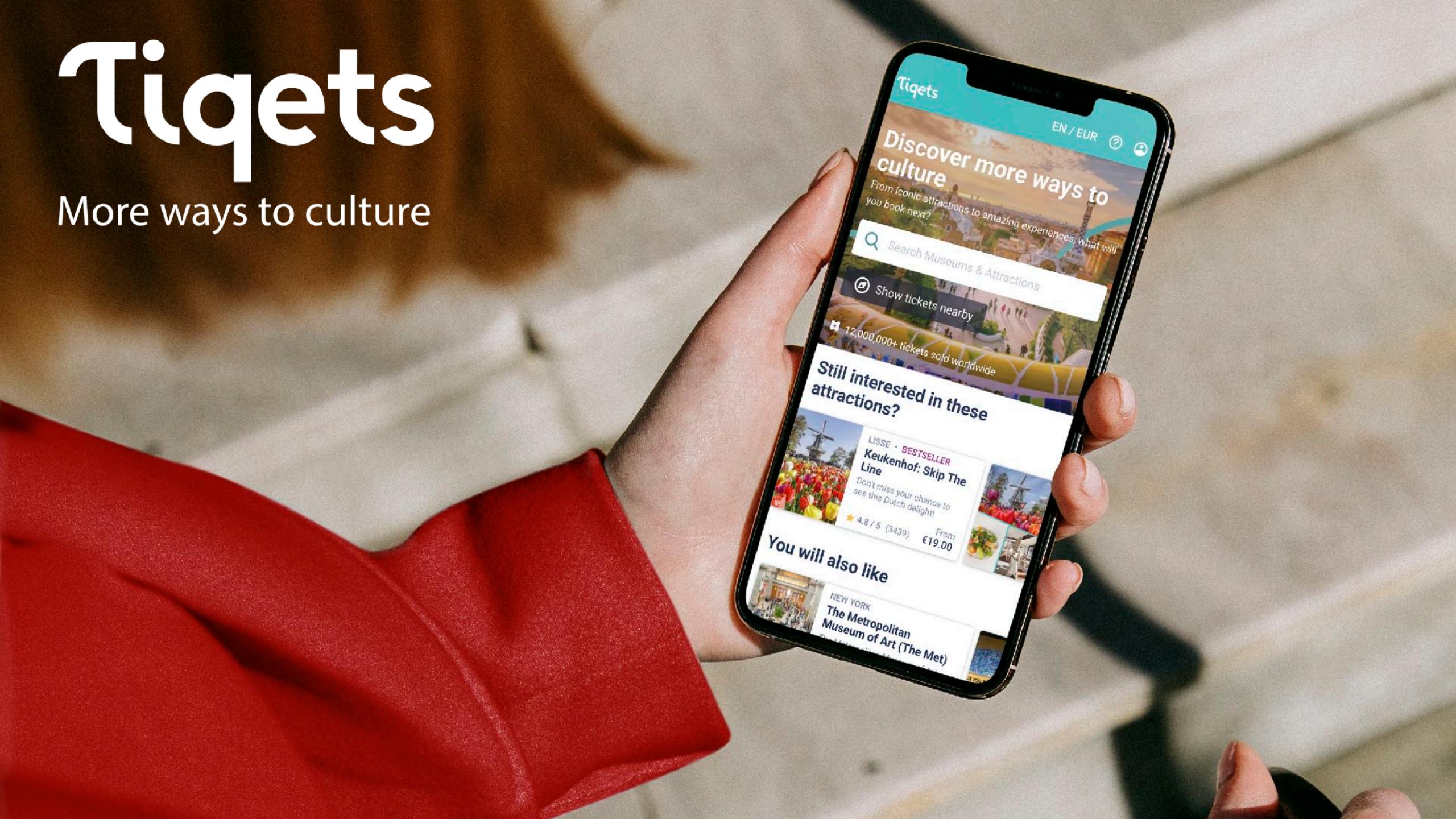


The plan ... and how it fails



Òscar Vilaplana

Technical Product Manager at Tiqets



1

Evaluation

Evaluation

Measure systematically the quality of your LLM pipeline to drive **systematic improvement**.

- Accuracy
- Relevance
- Coherence

- Safety
- Customer satisfaction
- Conversion



Drive systematic improvement

Understand what the LLM gets wrong

- In what contexts?
- How severe & frequent?

Fixed a problem: what's the impact

- Did it help? How much?
- What side effects did we introduce?
- How do we prevent regression?

New model: should we switch?

- What's the impact?
- What does it do better / worse?



2

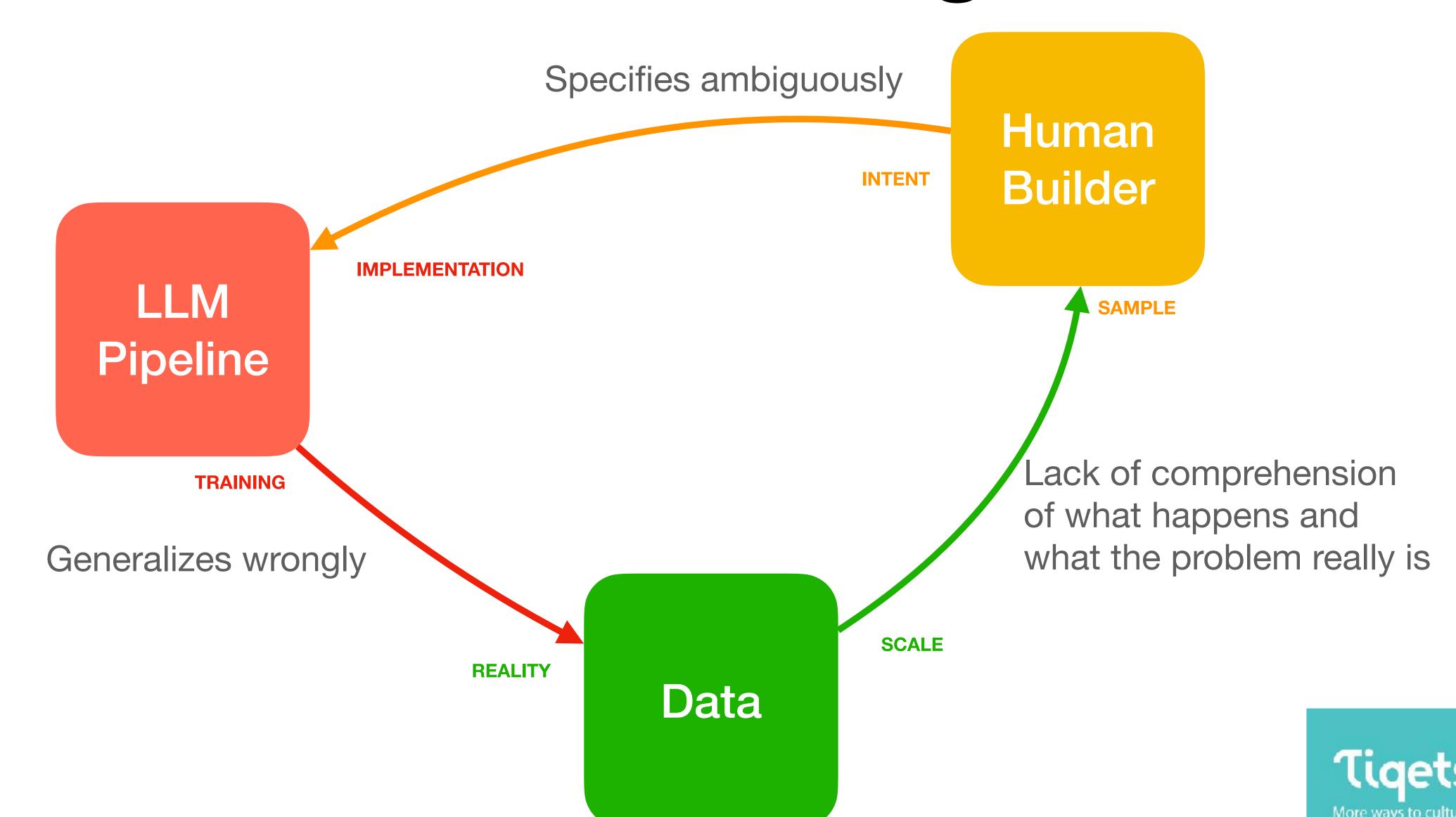
Pre-mortem

What went wrong?

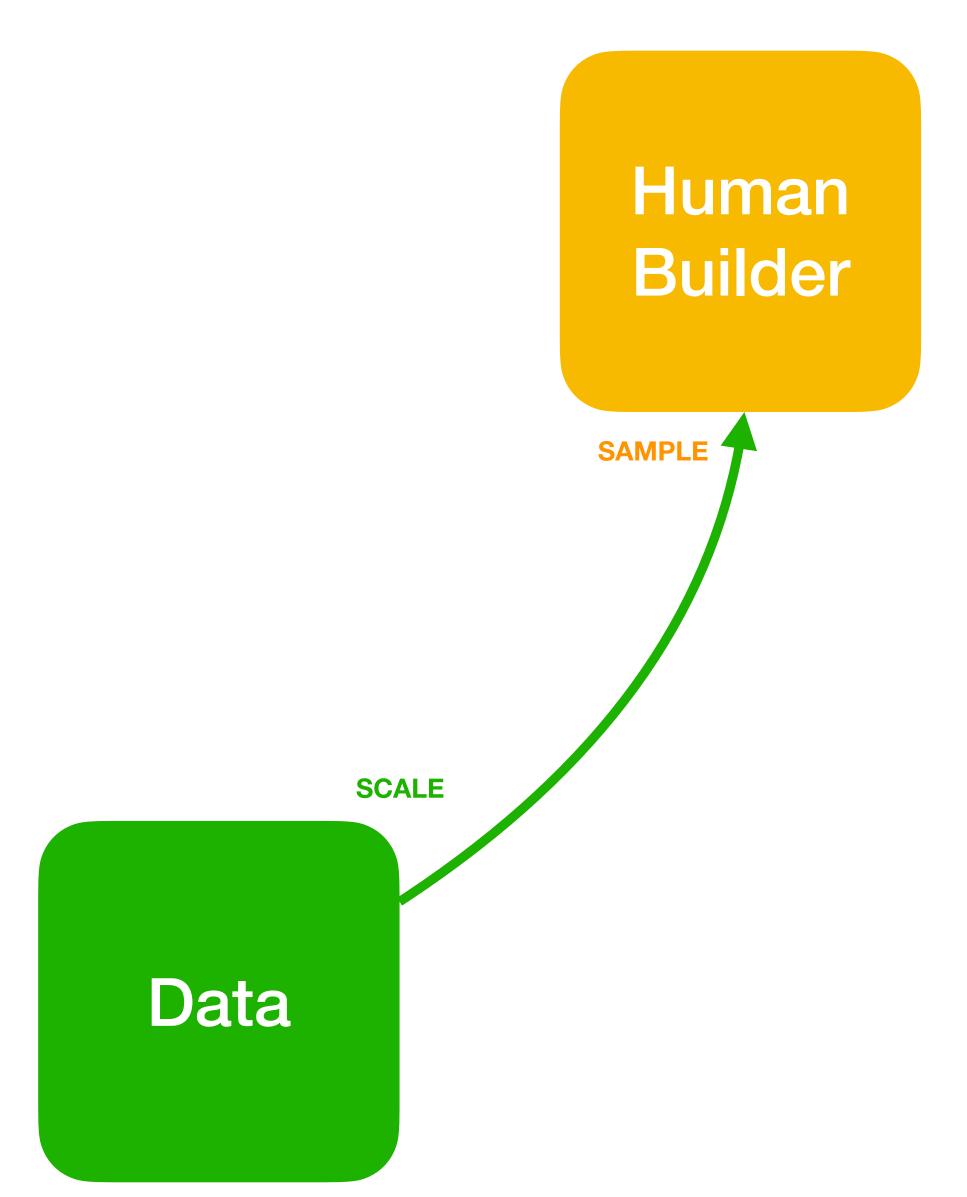
- How am I supposed to know?
- What were the inputs & outputs?
- What were you trying to do?
- What did you think the problem was?
- What did you tell the LLM to do?
- What did you think you had to tell the LLM?
- What did the LLM understand?



What went wrong?



Comprehension



- The problem you're trying to solve
- What happens. How does it go wrong
- What edge cases
 - What goes wrong
 - What user behaviors (vs our assumptions)
- What scenarios
- What patterns. What distribution

But you can't manually inspect all possible interactions between users and your LLM pipeline.



Specification



- What do you actually want the system to do?
- How do you specify your domain expertise and context?
- What does "good enough" mean in your domain?

- What does success look like?
- Which edge cases matter?
- What trade-offs are OK with?
- What failure modes are acceptable?



Generalization

 Generalizing the specification to every possible new situation is inherently difficult.

• Perfect prompts can't fix fundamental model limitations.

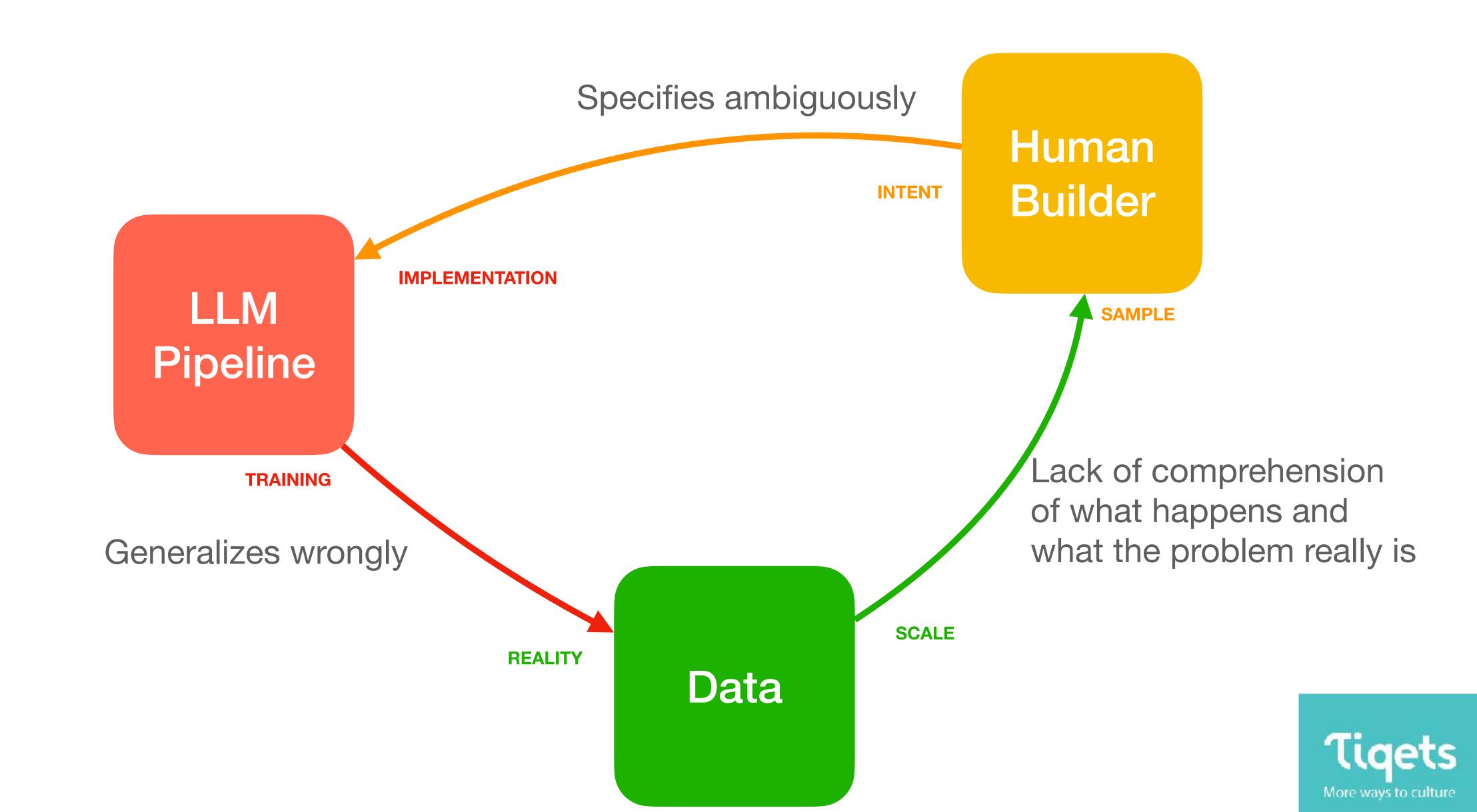
 You may need to: decompose complex tasks, fine-tune for your domain, or add retrieval augmentation.

Data





TRAINING



3

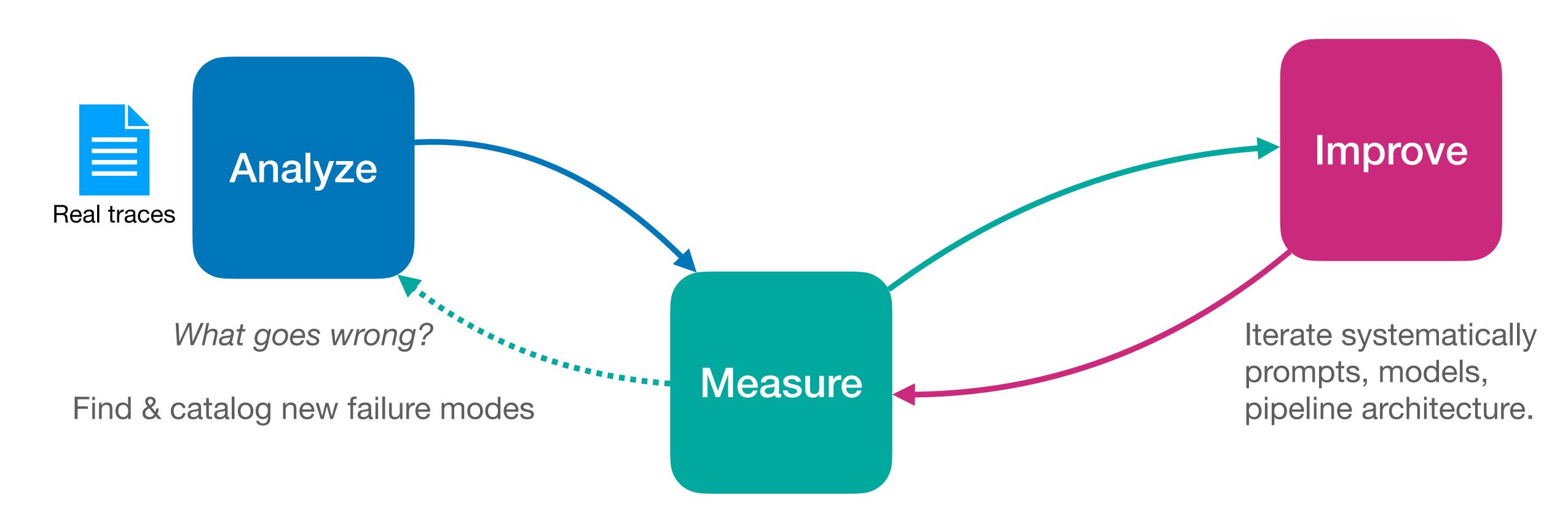
Automated Evaluation

Discovery-first evaluation

- 1. Discovery: What can go wrong?
- 2. Evaluation: How often does each thing go wrong?
- 3. Improvement: Did our fixes work?



Analyze, Measure, Improve

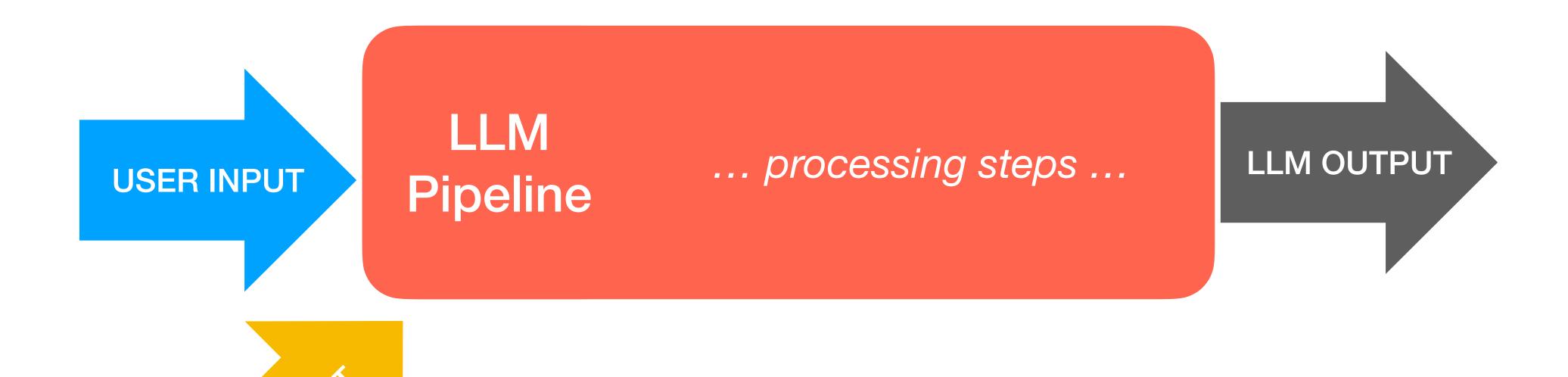


How often does each failure happen?

Did our improvements work?

Tiqets
More ways to culture

Trace: Complete Interaction Record



- What the user asked
- What data was available (conversation history, tools, other data)
- What the system did (processing steps, metadata)
- Generated response



Example Trace

Trace ID: groningen-B5ABD7EF-DE3E-4423-A8A4-83A9C98DF026

USER INPUT

User: "I'm gonna be around Groningen in the fall for a weekend, and I'll rent a bike. Can you suggest me things to do?"

Timestamp: 2025-11-09 14:32:45 UTC

Session ID: session_xyz789
User ID: anonymous_user_001

CONTEXT

Conversation History

First interaction in session (no prior context)

User Profile Data

- · Location: Detected IP region (Netherlands)
- · Language: English
- Previous interactions: None

Extracted Intent Parameters

city: "Groningen"
season: "fall"
duration: "weekend"

transport: "bike"

intent: "tourism_recommendations"

PROCESSING STEPS

Step 1: Query Classification

Component: Intent Classifier Action: Classify user intent

Result: tourism_recommendation (confidence: 0.94)

Duration: 45ms

Step 2: Entity Extraction

Component: NER Model

Extracted: Location=Groningen, Season=fall, Transport=bike

Duration: 67ms

Step 3: Knowledge Retrieval

Component: Tourism Database RAG

Query: "bike-friendly activities Groningen autumn"
Retrieved: 8 relevant activities, 3 bike routes

Duration: 234ms

Step 4: Weather API Call

Component: Weather Service

Query: Groningen weather forecast for current fall period

Result: Mild temps (12-16°C), occasional rain

Duration: 145ms

Step 5: Response Generation

Component: LLM (GPT-4)

Prompt: "Generate personalized tourism recommendations..."

Tokens: Input: 456, Output: 324

Duration: 1.2s

Step 6: Safety & Quality Check

Component: Content Filter

Checks: Toxicity, accuracy, completeness Result: PASSED (all checks green)

Duration: 89ms

LLM OUTPUT

Tourism Bot: Great choice! Groningen is perfect for cycling in fall. Here are my recommendations for your weekend: ♣ Bike-Friendly Highlights: • Explore the historic city center (Grote Markt, Martinitoren) • Cycle to Noorderplantsoen park - beautiful autumn colors • Visit the Groninger Museum (bike parking available) • Drop by aiLand and learn everything about Al! • Take the scenic route along the Reitdiep canal ♣ Fall-Specific Activities: • Saturday morning farmers market at Grote Markt • Autumn photography at Prinsentuin gardens • Cozy cafés for warming up: Café de Sleutel, De Pintelier ♠ Weather Tip: Pack a light rain jacket - temperatures around 12-16°C with possible showers. ♣ Recommended Route: City center → Noorderplantsoen → Museum → Prinsentuin (total: ~15km, perfect for a Saturday!) Would you like specific opening hours or more details about any of these suggestions?

Response Generated: 2025-11-09 14:32:47 UTC

Total Latency: 1.8 seconds Confidence Score: 0.87

Cost: \$0.0023 (tokens + API calls)

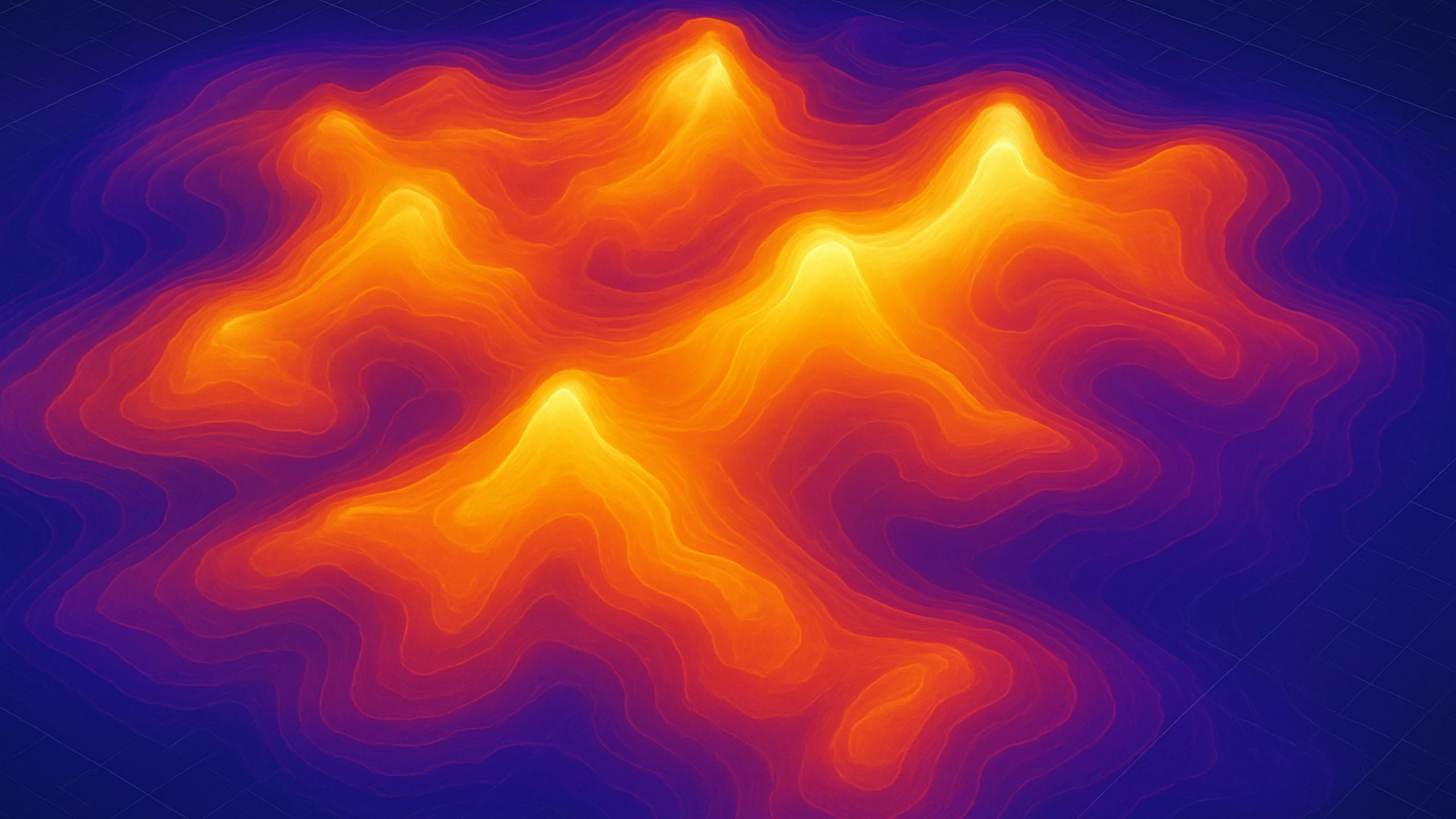


Failure Mode: How exactly did it go wrong?

A failure mode is a **specific** way in which your LLM system **failed** to do what you wanted.

- Hallucination
- Context loss: misses key information provided
- Failed to follow instructions
- Bias (from prompt, training)
- etc.





Failure Mode: How exactly did it go wrong?

A failure mode is a **specific** way in which your LLM system **failed** to do what you wanted.

- Hallucination
- Context loss: misses key information provided
- Failed to follow instructions
- Bias (from prompt, training)
- etc.



Automate to Amplify Insight

To automate evaluation, we need **insight** on what's going wrong.

Automated evals with insight won't be very helpful.

- It can only find generic failure modes.
- Lacks domain and product context: what matters for your use case?
- It has its own bias.

Automation can amplify insight. You need the insight to start with.

```
> llm_judge("Find problems in this tourism
recommendations response", trace)

"Response could be more concise"
# Too generic

> eval_transport_suggestion(trace)
```

Bike or public transport is more convenient.

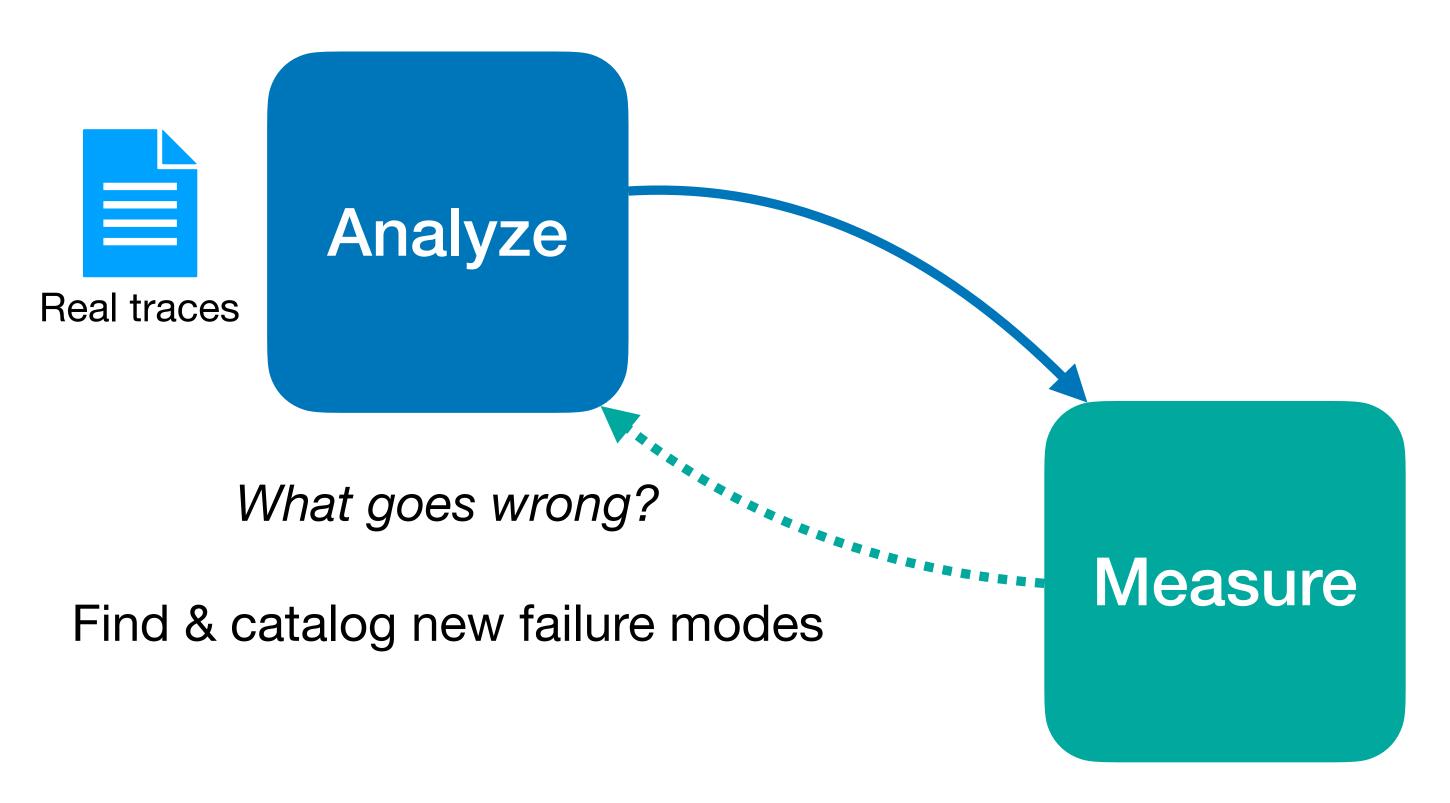
"Suggested city driving"

Parking in the city is expensive.

4

Error Analysis

Error Analysis



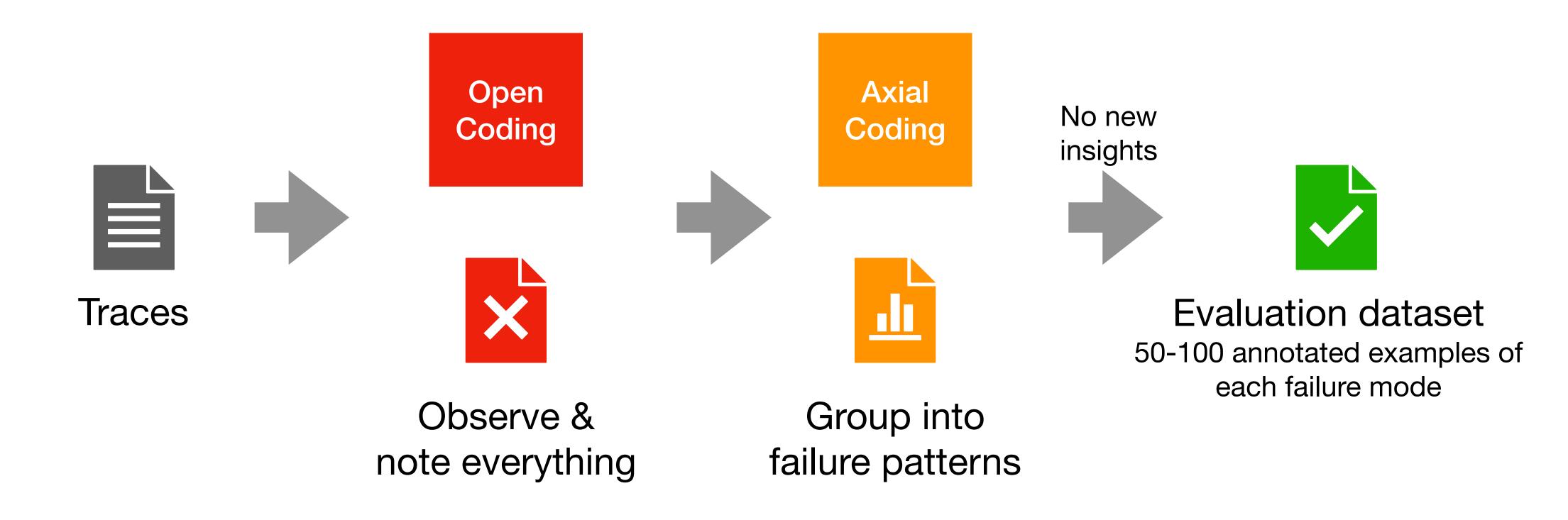
How often does each failure happen?

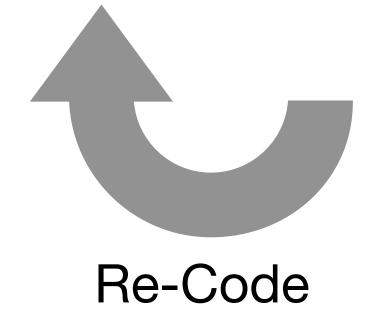
Did our improvements work?

Qualitative insights → Quantitative metrics



Error Analysis







Failure modes

Short Code	Detail
Didn't check availability	Recommended Anne Frank House tickets for tomorrow without checking sold-out status
Didn't check price	Quoted Keukenhof entry at €15 when current price is €19
Didn't check opening times or availability	Suggested 9 PM visit to Rijksmuseum (closes at 5 PM)
Conflicting reservations	Booked overlapping tours: 10 AM canal cruise and 10:30 AM Rijksmuseum tour
Incorrect location match	Recommended Rotterdam attractions when user asked about "Dam Square area"
Wrong travel time	Suggested visiting Giethoorn and Zaanse Schans in same 2-hour window
Wrong travel instructions	Directed to take metro to Volendam (no metro connection exists)
Missed weather context	Recommended Vondelpark picnic during December storm warning
Stale calendar data	Promoted King's Day celebrations on April 28th (moved to 26th)
Switched language	Responded in Dutch to an English query
Ignored accessibliity context	Suggested windmill climb to wheelchair user without warning
Ignored price filters	Returned €200/person dinner recommendations for "budget-friendly" request
Wrong time of year	Suggested Christmas markets in July
Capacity oversight	Recommended intimate canal boat for 25-person corporate group
Preference blindness	Suggested art museums to user who explicitly said "no museums"
Tourist vs local confusion	Explained what a stroopwafel is to an Utrecht local
Too many options	Listed 47 activities for simple "what to do this afternoon" query

Synthetic Queries

1. Decide what **dimensions** make sense for **your product**.

2. Generate scenarios based on your dimensions

- 3. Ask an LLM to generate queries for each scenario + examples
- 4. Review, discard unrealistic queries
- 5. Call your product and obtain the traces

eg: intent, query complexity, type of traveler, time sensitivity, ...

intent: travel recommendations
query complexitiy: simple

type of traveler: family with children

time sensitivity: planning

intent: cheap tickets

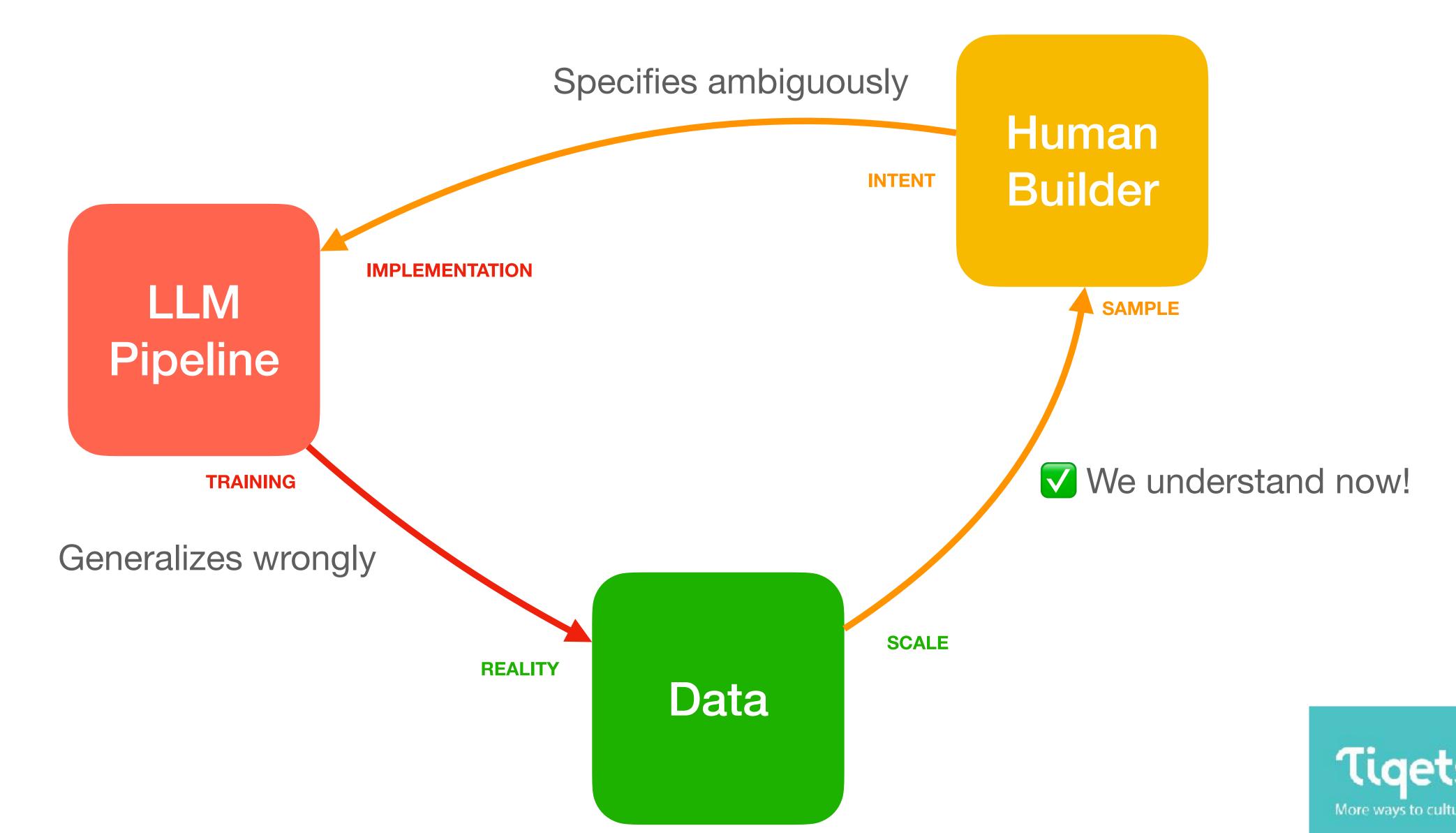
query complexitiy: complex type of traveler: solo, young time sensitivity: last minute



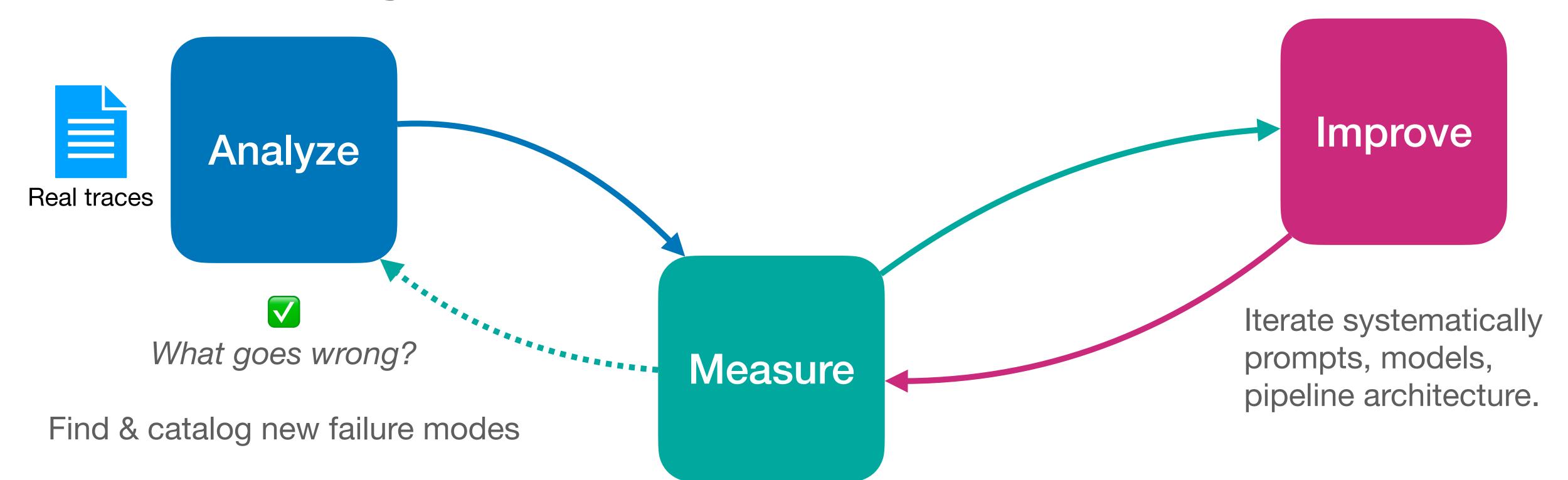
5

Automation

Specification vs Generalization



Analyze, Measure, Improve



How often does each failure happen?

Did our improvements work? .

Qualitative insights → Quantitative metrics



Automate to Amplify Insight

To automate evaluation, we need **insight** on what's going wrong.

Automated evals with insight won't be very helpful.

- It can only find generic failure modes.
- Lacks domain and product context: what matters for your use case?
- It has its own bias.

Automation can amplify insight. You need the insight to start with.

```
> llm_judge("Find problems in this tourism
recommendations response", trace)

"Response could be more concise"
# Too generic
```

```
> eval_transport_suggestion(trace)
```

```
"Suggested city driving"

# Parking in the city is expensive.

# Bike or public transport is more c
```



Evaluators

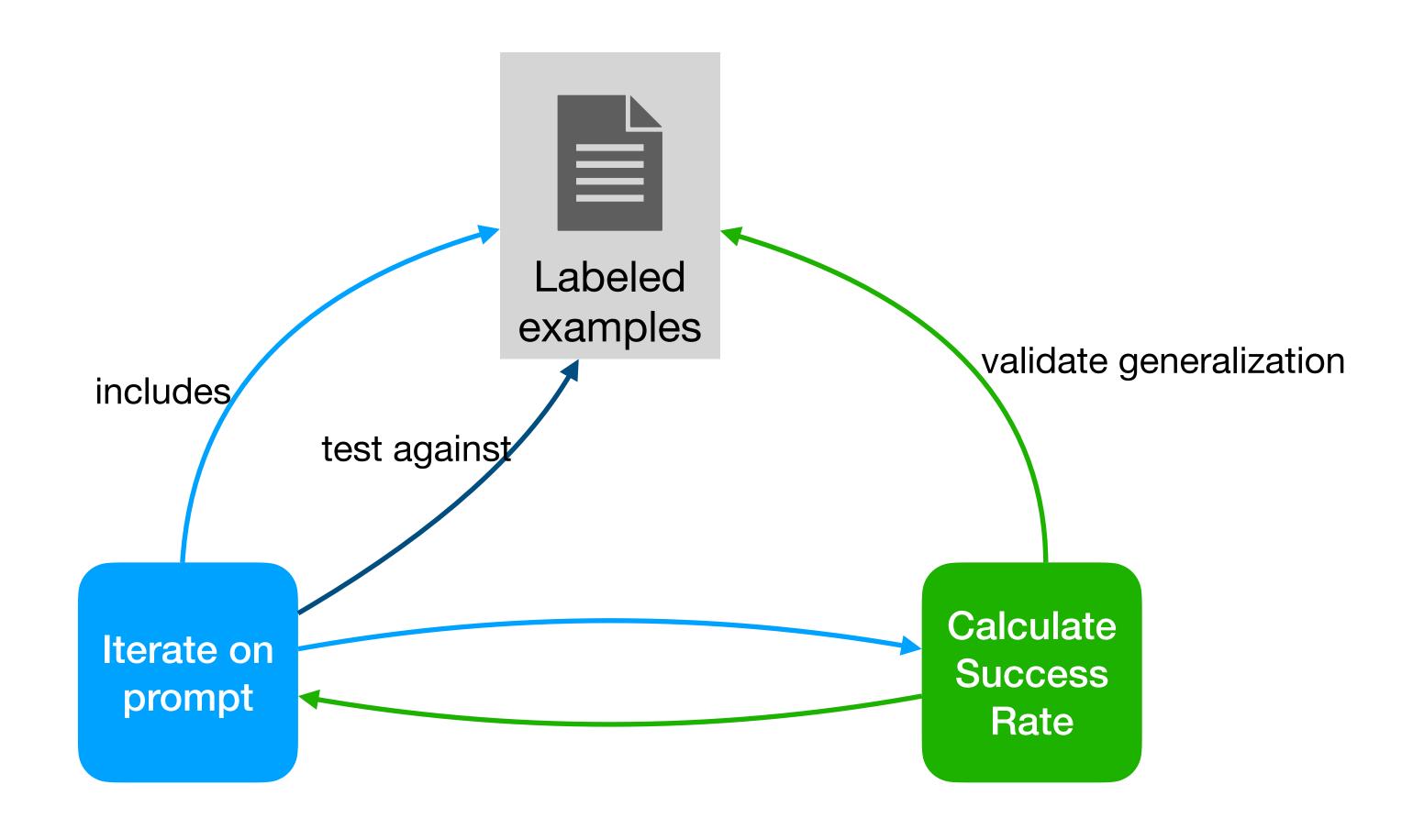
```
Algorithmic Evaluator

def eval_valid_json(trace: Trace) -> bool:
    """The output should be a valid JSON"""
    try:
        json.loads(trace.output)
        return True
    except json.JSONDecodeError:
        return False
```

```
def eval_time_sensitivity(trace: Trace) -> bool:
    """The output should take into account
    the user's time sensitivity (planner vs last-minute)."""
    llm_response = call_llm(prompt=prompts.TIME_SENSITIVITY, trace=trace)
    return llm_response.result == "pass"
```



LLM-as-Judge





Training / Dev / Test Split



20%

Few-shot examples

Include in the prompt



40%

Iterate on LLM changes



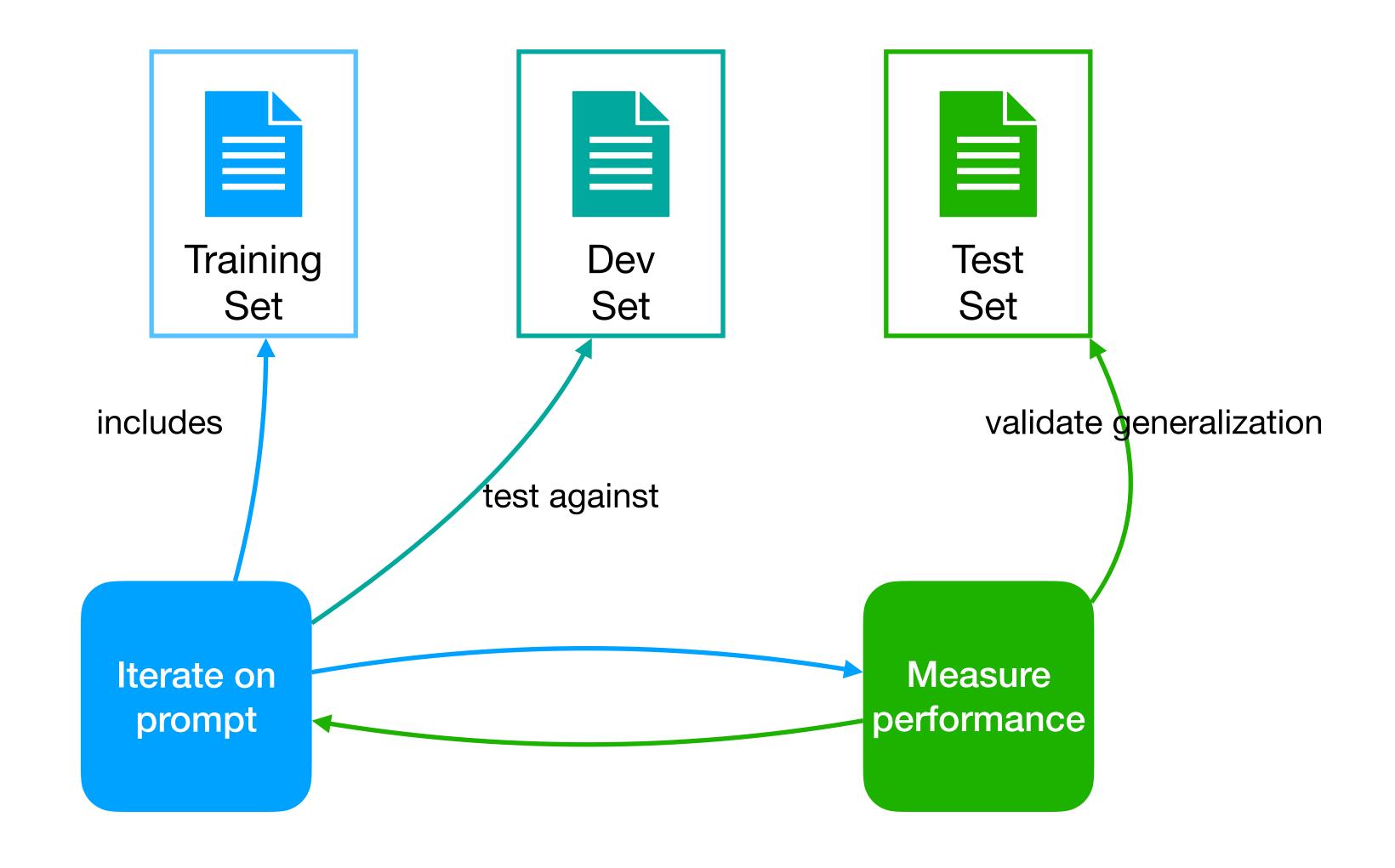
Test

40%

Measure performance



LLM-as-Judge





LLM-as-Judge Prompt

- Focus on only one failure mode.
- Give it a very clear task.
- Explain precisely what it means to pass and what it means to fail.
- Give it few-shot examples of both pass and fail. Choose especially interesting and difficult examples.
- Get output in a structured format that includes reasoning and pass / fail.

You are an expert evaluator.

Evaluate whether the Tourist Attractions Assistant responds with travel suggestions relevant to the user.

Responses should offer maximum 10 concrete options when the query is specific, or ask clarifying questions when the query is too ambiguous.

FAIL IF the agent:

- List more than 10 alternatives.
- Gives generic options to an ambiguous query.

PASS IF the agent:

- Lists 10 alternatives or fewer.
- Asks relevant clarifying questions to an ambiguous query.

```
< Few-Shot Examples, including query and desired output)>
```

```
Output format: JSON
{
    "reasoning": "Brief explanation of option count and appropriateness",
    "answer": "Pass" | "Fail"
}
```



DSPy Programming—not prompting—LMs

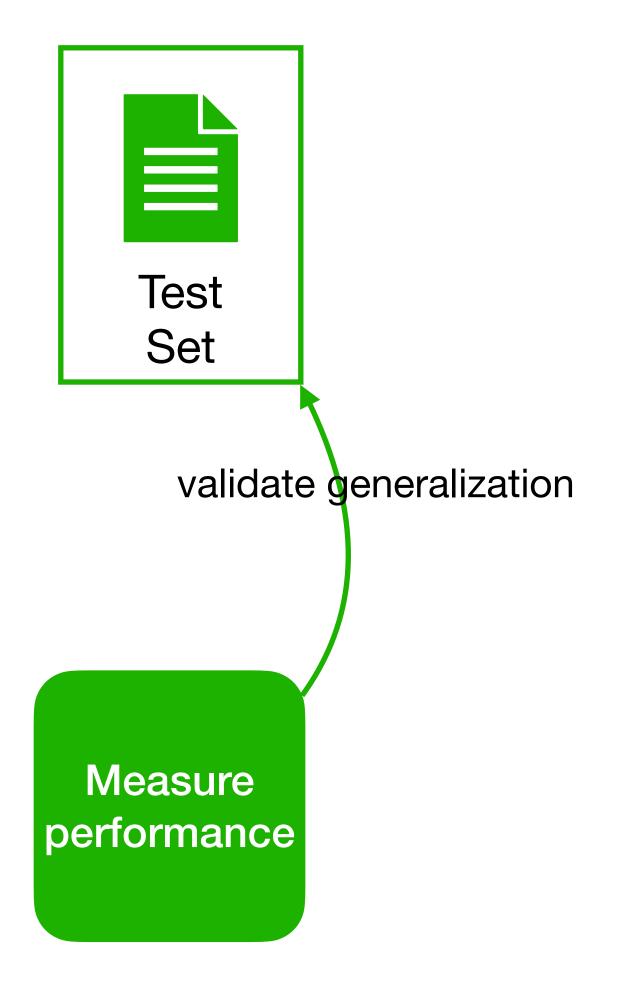
Measure performance

How much does the LLM as judge agree with your judgment?

- Calculate True Positive Rate (TPR). How often does the judge agree with you on a pass?
- Calculate the true negative rate (TNR): how often does the judge agree with you on a fail?

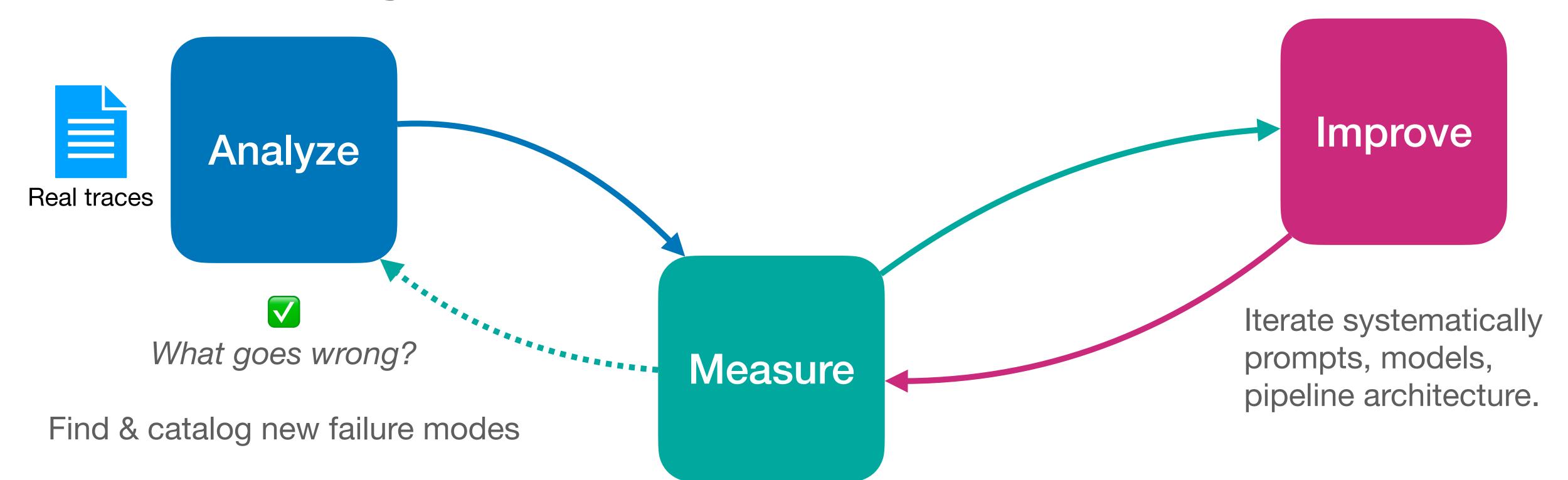
When there is a disagreement, inspect the case and see why the judge might have gotten it wrong. What should change in the prompt? Do you need better examples?

Repeat until you get enough agreement. (depending on your product: 70%? 90%?)





Analyze, Measure, Improve



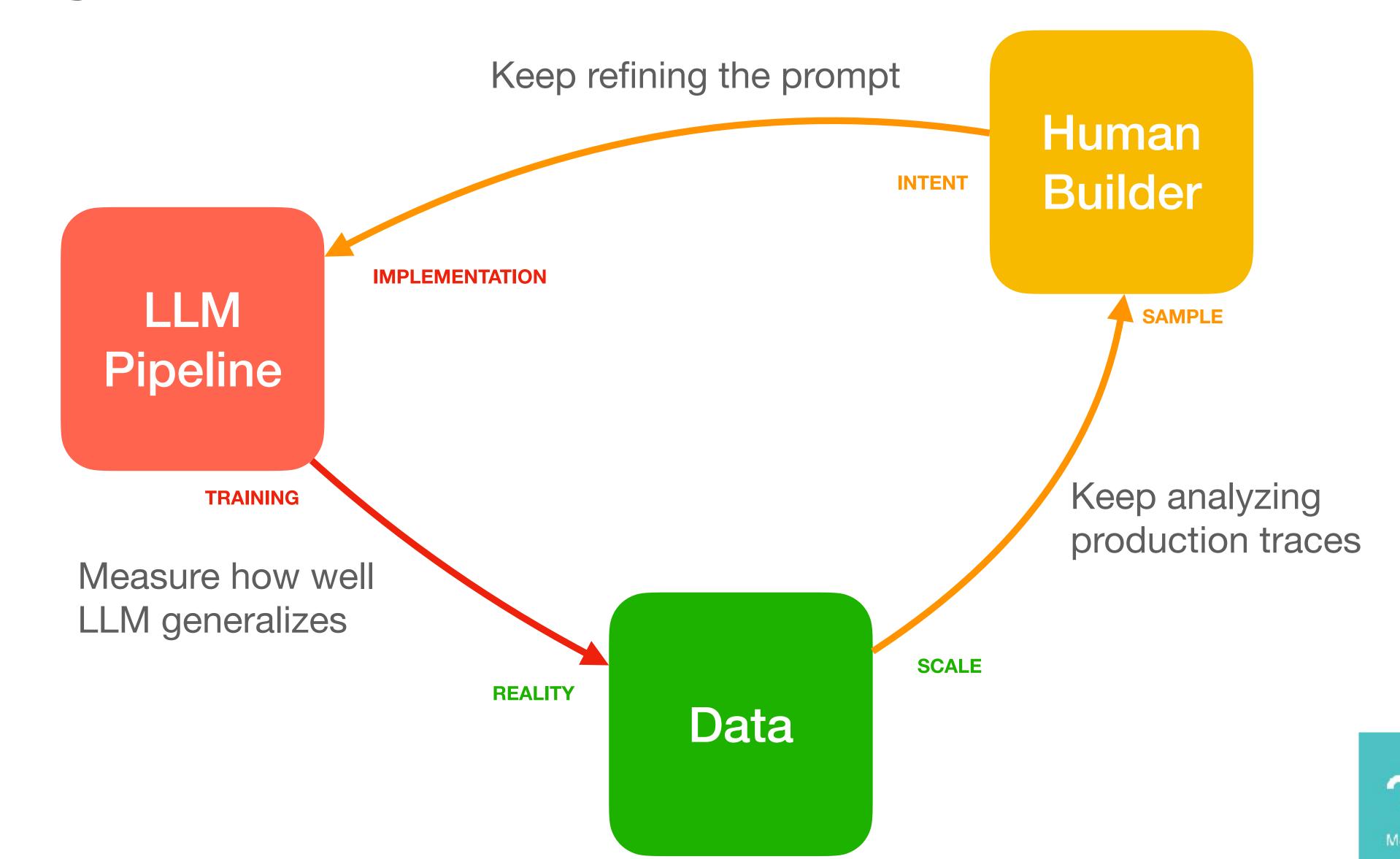
How often does each failure happen?

Did our improvements work? .

Qualitative insights → Quantitative metrics



Systematic Improvement



Questions?





Thank you!



