Generative AI: Should We Say Goodbye to Deterministic Testing?

Dean Wampler, Ph.D.
The AI Alliance and IBM
dwampler@thealliance.ai

aialliance.org

deanwampler.com/talks

Outline

- First, about the AI Alliance
- How non-deterministic AI affects testing
- So, what should we developers do?
- Thinking about a new perspective

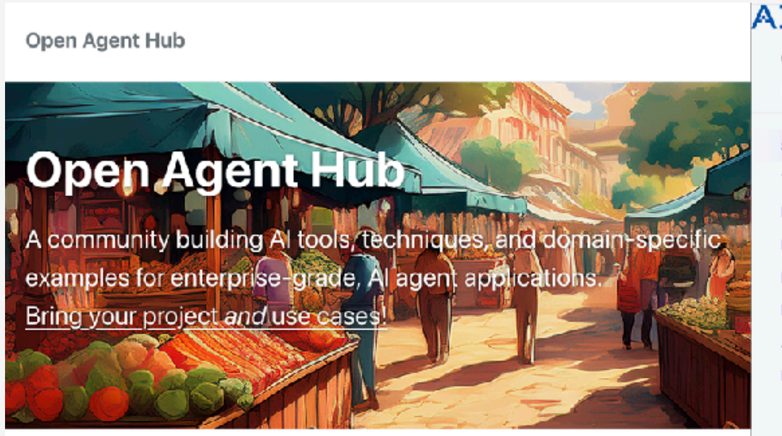
This isn't a "problem solved!" talk. I'll describe the problem and outline potential solutions.

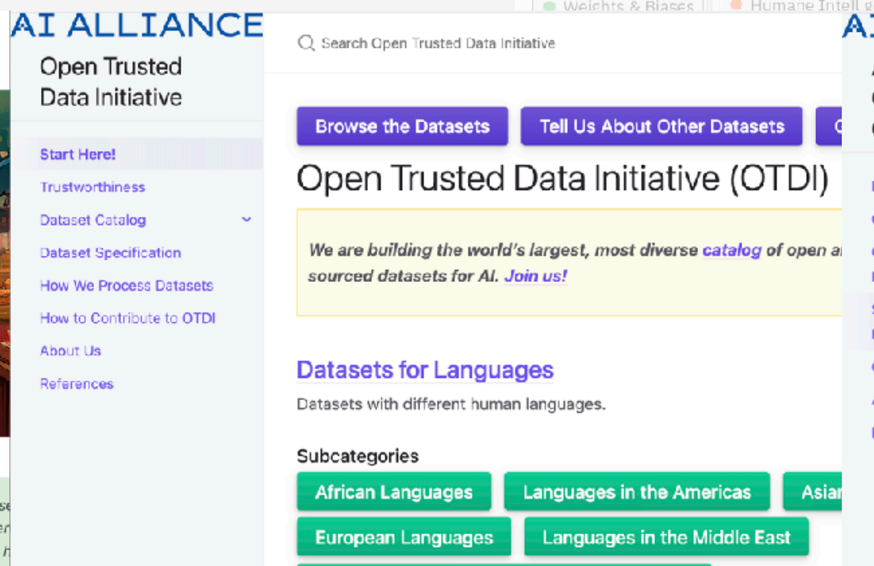
AI ALLIANCE

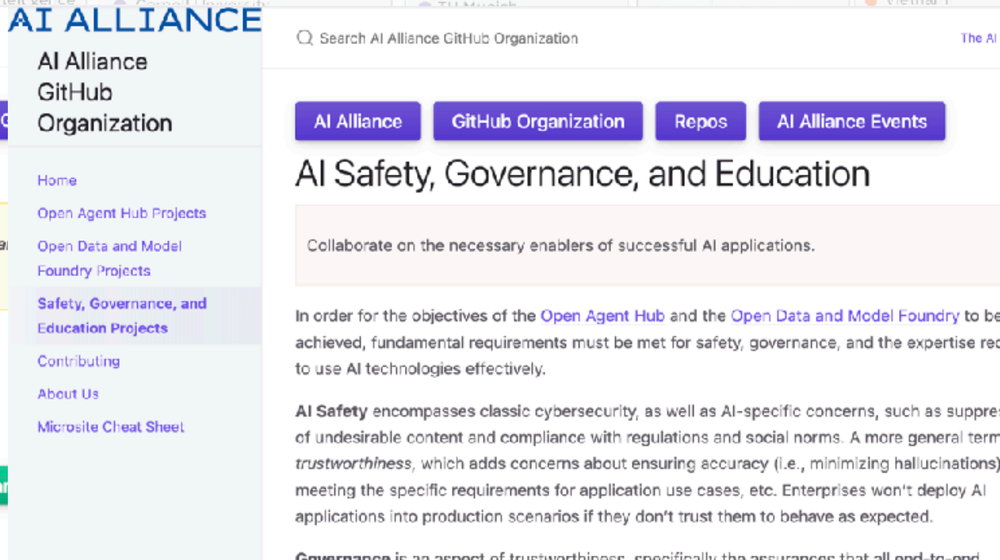
195+ organizations in 25+ countries accelerating open innovation and adoption of AI

These project areas:
the-ai-alliance.github.io/

1. Agents
2. Data & Models
3. Safety & Governance







Search Testing Generative AI Agent Applications

The AI Alliance

Join This Project | GitHub Repo

Testing Generative AI Agent Applications

(Previous Title: *Achieving Confidence in Enterprise AI Agent Applications*)

I am an enterprise developer; how do I test my AI agent applications??

*I know how to test my traditional software, which is **deterministic** (more or less...), but I don't kno*
*applications, which are uniquely **stochastic**, and therefore **nondeterministic**.*

Welcome to the **The AI Alliance** project to advance the state of the art for **Enterprise Testing of G**
Applications. We are building the knowledge and tools you need to achieve the same testing *confi*
applications that you have for your traditional applications.

Note:

This site isn't about using AI to generate conventional tests (or code). You can find many online resources about that topic. Instead, this site focuses on the problem of how to do testing of any kind when an application contains generative AI components, given the nondeterminism they introduce.

The Challenge We Face

We enterprise software developers know how to write Repeatable and Automatable tests. In particular, we rely on Determinism when we write tests to verify expected Behavior and to ensure that no Regressions occur as our code base evolves. Why is determinism a key ingredient? We know that if we pass the same arguments repeatedly to *most* Functions (with some exceptions), we will get the same answer back consistently. This property enables our core testing techniques, which give us ***the essential confidence*** that our applications meet our requirements, that they implement the Use Cases our customers expect. We are accustomed to unambiguous *pass/fail* answers!

Problems arise when we introduce Generative AI Models, where generated output is inherently Stochastic, meaning the outputs are governed by a probability model, and hence *nondeterministic*. We can't write the same kinds of tests now, so what alternative approaches should we use instead? The problems are compounded when we have applications built on Agents, each of which will have some stochastic behavior of its own, if it encapsulates a generative model.

In contrast, our AI-expert colleagues (researchers and data sci
stochastic model responses and to assess how well the models perform against particular objectives. For example, a model

An AI Alliance project I lead to work on AI testing tools and techniques.

https://the-ai-alliance.github.io/ai-application-testing/

AI ALLIANCE

io/

In order for the objectives of the Open Agent Hub and the Open Data and Model Foundry to be achieved, fundamental requirements must be met for safety, governance, and the expertise re to use AI technologies effectively.

AI Safety encompasses classic cybersecurity, as well as AI-specific concerns, such as suppre of undesirable content and compliance with regulations and social norms. A more general term *trustworthiness*, which adds concerns about ensuring accuracy (i.e., minimizing hallucinations) meeting the specific requirements for application use cases, etc. Enterprises won't deploy AI applications into production scenarios if they don't trust them to behave as expected.

Governance is an aspect of trustworthiness, specifically the assurances that all end-to-end processes used to create all AI application components are secure, licensed for use, etc. AI mo are created with data; they are mostly data themselves. Hence, models, like data, need to be governed.

Finally, **Education** addresses the needs where organizations struggle to learn all the things the to know in order to use AI safely and effectively. Not only has AI introduced new tools and tech to software application development, it has fundamentally altered some of the ways software w for example, introducing *stochastic* behaviors as core aspects of application features, where previously *deterministic* behaviors were the norm. Most AI Alliance projects have dual missions only to innovate and create, but to educate.

The following projects address these concerns.

Links	Description
The AI Trust and Safety User Guide	A

AI ALLIANCE

Join us!

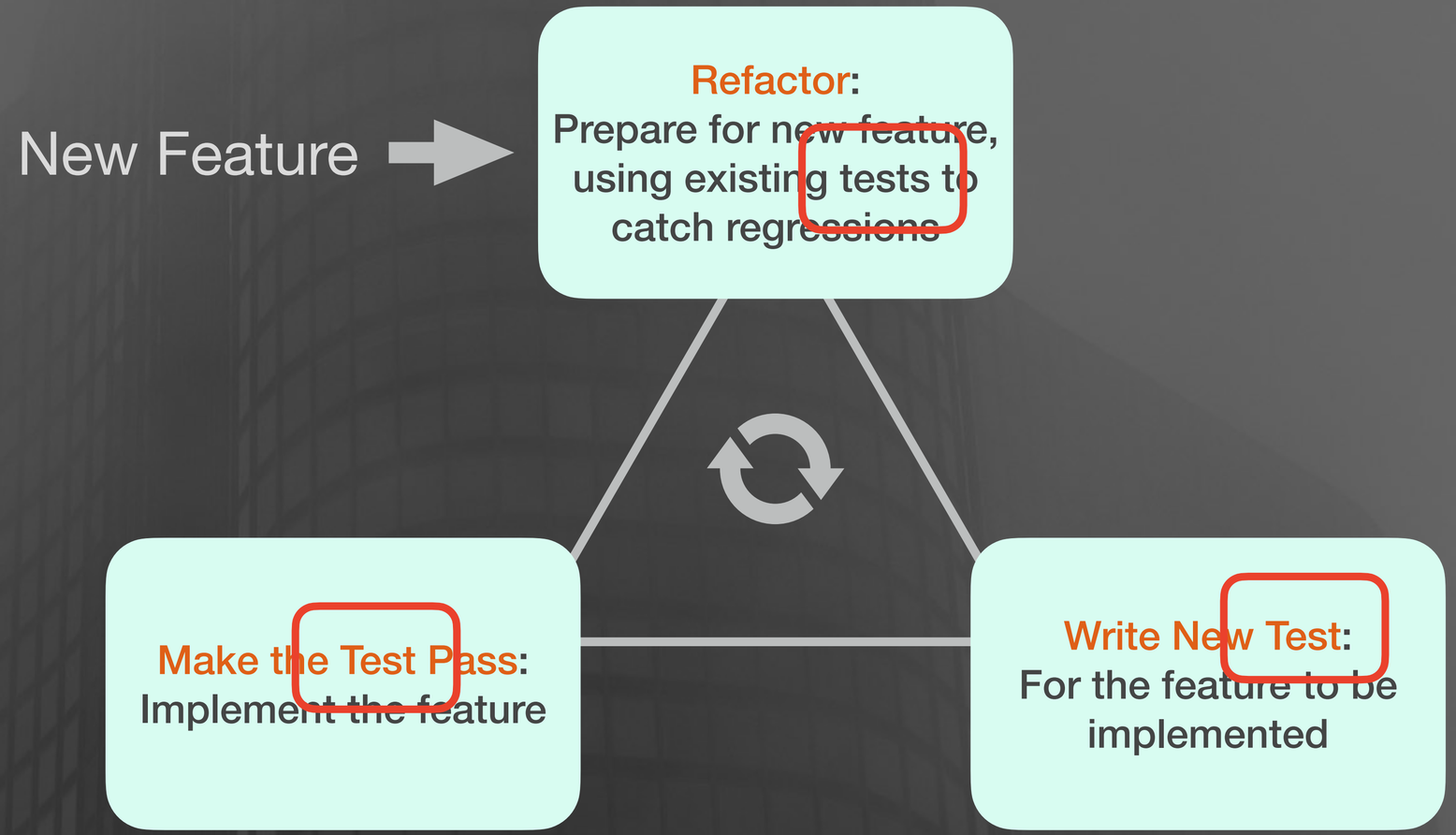
- aialliance.org

bsky.app/profile/aialliance.bsky.social

linkedin.com/company/the-aialliance/


AI is non-deterministic.
How does that affect testability?

Remember the TDD‡ loop?



Testing is the foundation of this process!

‡ Test-Driven Development

What Do Developers Expect?

Developers expect software to be deterministic‡.
This helps ensure correctness, and reproducibility enables automation that catches regressions:

- The same input → the same output.
  - e.g., $\sin(\pi) = -1$
- The output changes? Something broke!

‡ Distributed systems break this clean picture.

What Do Developers Expect?

Develope inistic‡
behavior
reproduc
- The s
  - e.g.,
- The o roken!

Put another way, the determinism makes it easier to specify the system invariants. What should remain true before and after each step?

‡ Distributed systems break this clean picture.

What Do Developers Expect?

Functional Programming gave us property-based testing:

- E.g., QuickCheck, Hypothesis, ScalaCheck, …
- Hypothesis example:

```
@given(st.integers(), nonzero_integers, st.integers(), nonzero_integers)
def test_two_non_identical_rationals_are_not_equal_to_each_other(self, numer1, denom1, numer2, denom2):
    """
    Rule: a/b == c/d iff ad == bc
    This is a better test, because it randomly generates different instances.
    However, the test has to check for the case where the two values happen to be
    equivalent!
    """
    rat1 = Rational(numer1, denom1)
    rat2 = Rational(numer2, denom2)
    if numer1*denom2 == numer2*denom1:
        self.assertEqual(rat1, rat2)
    else:
        self.assertNotEqual(rat1, rat2)
```

https://github.com/deanwampler/tdd-hypothesis-example

What do we get with generative AI?

Generative models are stochastic‡:

- The same prompt → **different** output.
  - chatgpt(“Write a poem”) → insanity

“Insanity is doing the same thing over and over again and expecting different results.” — not Einstein

‡Stochastic : described by a random probability distribution, e.g., flipping a coin, rolling dice, measuring the temperature, …

What do we get with generative AI?

Generative models are stochastic‡:

- The same prompt → **different** output.
  - chatgpt("Write a poem") → insanity
- Without determinism, how do you write repeatable, reliable tests for AI apps?
  - Does that new model perform better or worse than the previous model?
  - Did any regressions in behavior occur?

What do we get with generative AI?

Generat

- The
  - cha
- With
  repe
  - Do
    tha
  - Dic

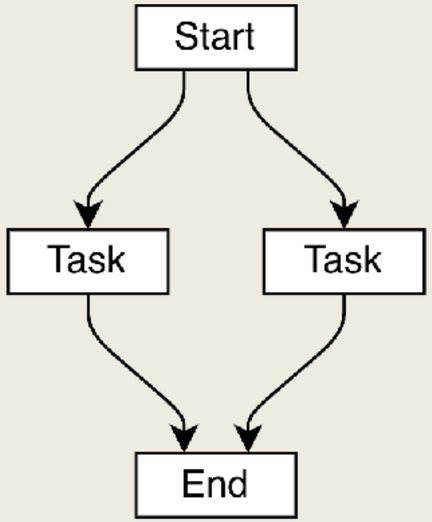
ite

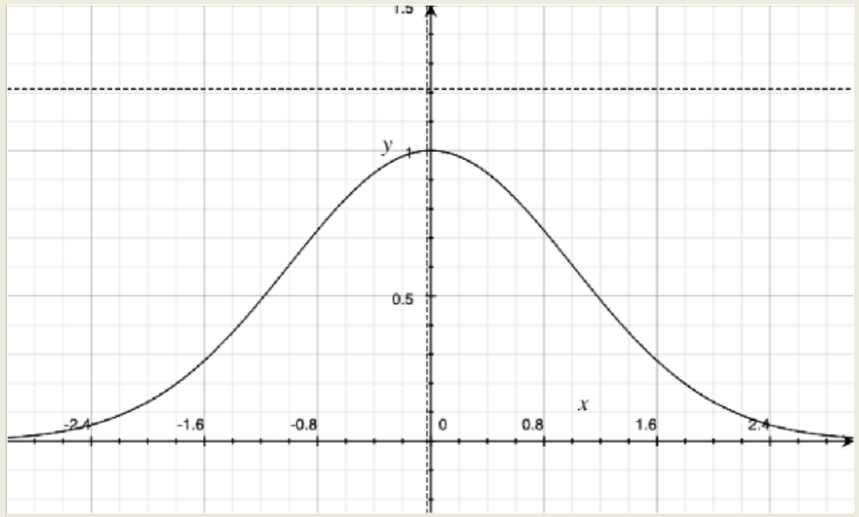
or worse

?

Put another way, the invariants are much less clear and therefore harder to define programmatically and enforce.

But Data Scientists and AI Experts Are Accustomed to Stochasticity





Deterministic Behavior

Stochastic Behavior

Software Developer

AI Expert

The Spectrum of Experience


So, what should we
developers do?

So, what should we developers do?

- Leverage what you already know about coupling and cohesion
- Use external tools for verification
- Scoped benchmarks - “unit benchmarks”
- Use an LLM as a judge
- Learn and use statistical techniques

AI ALLIANCE

Testing Generative AI Agent Applications

Search Testing Generative AI Agent Applications

The AI Alliance

https://the-ai-alliance.github.io/ai-application-testing/

Join This Project | GitHub Repo

Testing Generative AI Agent Applications

(Previous Title: *Achieving Confidence in Enterprise AI Agent Applications*)

I am an enterprise developer; how do I test my AI agent applications??

*I know how to test my traditional software, which is **deterministic** (more or less...), but I don't know how to test my AI agent applications, which are uniquely **stochastic**, and therefore **nondeterministic**.*

Welcome to the **The AI Alliance** project to advar
Testing of Generative AI Agent Applications. V
you need to achieve the same testing *confidence*
your traditional applications.

Note:

This site isn't about using AI to generate conve
online resources about that topic. Instead, this
testing of any kind when an application contain
nondeterminism they introduce.

The Challenge We Face

We enterprise software developers know how to write Repeatable and Automatable

An AI Alliance project I lead to:

- Develop new developer testing tools and techniques adopted from data science.
- Teach developers how to use them.

Coupling and Cohesion

Coupling and Cohesion

- The non-deterministic AI model isn't the whole application. (E.g., Agent architectures)
  - Wrap the model in a good API.
  - Use deterministic test doubles for it.
  - Test everything else like you normally do.

Coupling and Cohesion

- Writing a good API:
  - Engineer your prompts to constrain outputs.
  - Use tools like Pydantic-AI for type safety (example).
  - Select the Gen AI models that seem to work best with your tools.

Coupling and Cohesion

- Writing
  - Engine
  - Use to
  - Select t

outputs.

afety

to work

Thinking about types encourages you to find ways to constrain model queries such that the responses are more closely aligned with your goals.

Coupling and Cohesion

- Writing
  - Engine
  - Use too
  - Select t

outputs.

afety

(examp

to work best wi

Most AI-enabled apps won't be open-ended chatbots, but use AI to resiliently translate between human text and tool APIs, and translate tool-to-tool interactions, so we don't have to do that translation in code ourselves.

Coupling and Cohesion

- However, tried and true C&C techniques don't help us test the model input and output behaviors themselves, nor do they eliminate the non-determinism that is unavoidable in our acceptance tests‡.

‡ The integration tests that prove features are done.

Use external tools for verification

Use external tools for verification

- Are you asking a model to generate code?
  - Check it with a parser or compiler
  - Scan for security vulnerabilities
  - Check for excessive cyclomatic complexity
  - Check that only allowed third-party libraries and versions are used.
  - …

Use external tools for verification

- Are you asking a model to generate code?
  - Using TDD? If you ask for code that makes your hand-written tests pass, does the generated code allow the tests to pass?
    - (Example)

Use external tools for verification

- Are you
  - Using T
    your ha
    generat
    - (Exan

Currently, I don't think many models are very good at generating powerful tests, but they can do a reasonable job generating code to pass the tests.

Use external tools for verification

- Are you asking a model to do logic or reasoning?
  - Check it with a logic/reasoning engine
    - Or use that tool instead to create your logic!

Use external tools for verification

- Are you asking a model to do planning?
  - Check it with a planning engine
    - Or use that tool instead to create your plan!

Use external tools for verification

- Are you asking a model to generate possible chemicals or physical processes?
  - Try creating and testing the chemical in a lab.
  - Test the physical process with a simulator.
    - (Letting AI generate the "idea", then testing in a simulator may be cheaper than using the simulator to generate possible ideas.)

Use external tools for verification

One of the reason that Agents are so popular now is the recognition that models can't do everything well (or cheaply). So, complementing models with other tools provides the best results.

Scoped benchmarks - “unit benchmarks”

Scoped benchmarks - "unit benchmarks"

- Models are evaluated with benchmarks.
  - Use a large number of examples.
  - Typically cover a broad topic,
    - e.g., effective Q&A, detect hate speech, detect bias, measure throughput, …
  - Return a single measurement, usually 0-100%.

Scoped benchmarks - "unit benchmarks"

- Not the same thing as a developer "test".
- But can we adapt the idea for testing?
  - Use a very narrow scope.
  - Still use a lot of examples for higher confidence.
  - Return a single measurement, usually 0-100%.
    - But at what threshold do you "pass"??

Scoped benchmarks - "unit benchmarks"

- Example: SQL queries generated from text.
  - Build a Q&A dataset that uses logged queries (expected answers) with appropriate human prompts (the queries).
  - Each unit benchmark might focus on one specific kind of common query.

This is also an example of using a model to translate between text and an "API".

Use an LLM as a judge

Use an LLM as a judge

- You have probably chosen a small model for production, because it costs less to use.
- Use a bigger, smarter for test runs to "judge" responses.
  - You'll call it less often, so the cost won't be as much of an issue.

Need data for your unit benchmarks? Use a big model to synthesize data!

Use an LLM as a judge

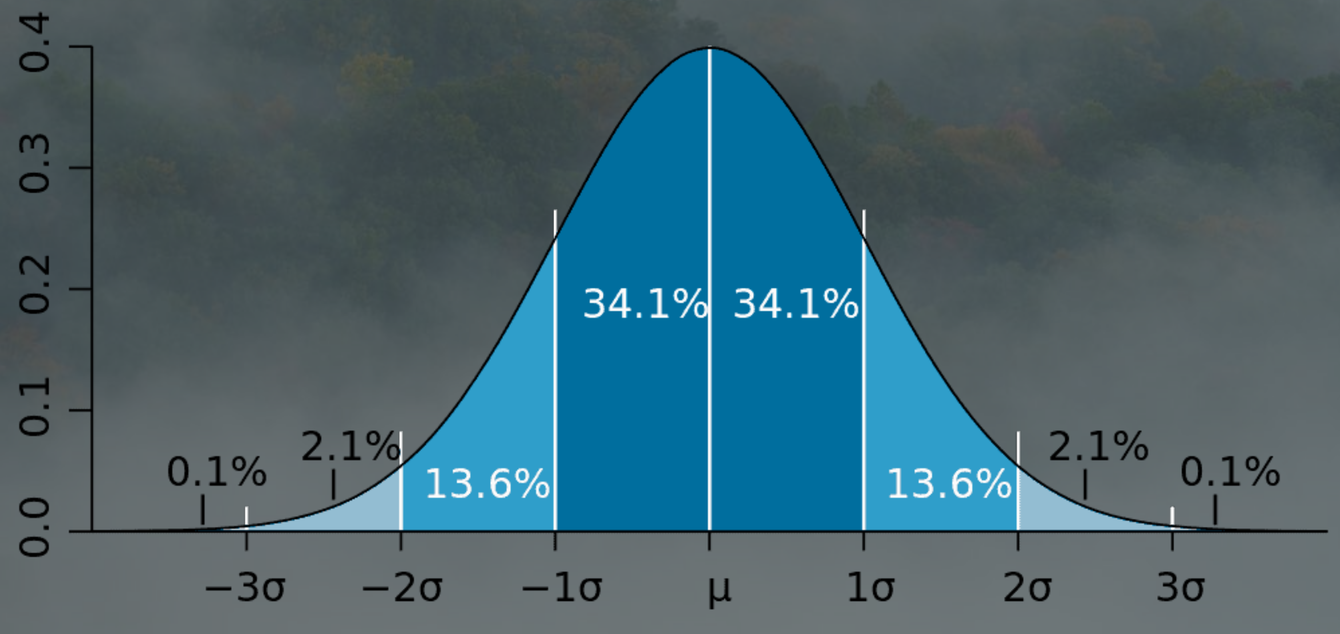
- It can work like this:
  - A test sends a query to the model or app.
  - The query and the response are sent to a larger model with the question, “Is this a good response for this query? Answer yes or no, and if no, provide an explanation.”
  - Fail the test if the answer is no.
  - Use the explanation to debug.

Understand and leverage statistics

Understand and leverage statistics

- Scientists are accustomed to using statistics to analyze probabilistic phenomena.
  - E.g., a potential discrepancy between theory and experiment must be > five sigma.

"A five-sigma level translates to one chance in 3.5 million that a random fluctuation would yield the result."
Wikipedia



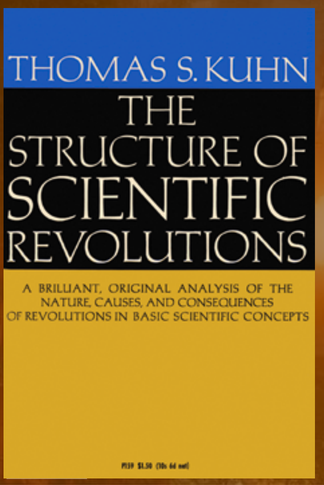
Understand and leverage statistics

- Classifier models sometimes return a confidence level, i.e., how much they believe they are returning the correct classification.
  - "Adding Error Bars to Evals: A Statistical Approach to Language Model Evaluations"
    - https://arxiv.org/abs/2411.00640

Thinking about a new perspective

Thinking about a new perspective?

- The Structure of Scientific Revolutions
  - We prefer to adapt our current theory to accommodate new data, rather than discard the theory and start over.
    - But sometimes, we need to restart from first principles.



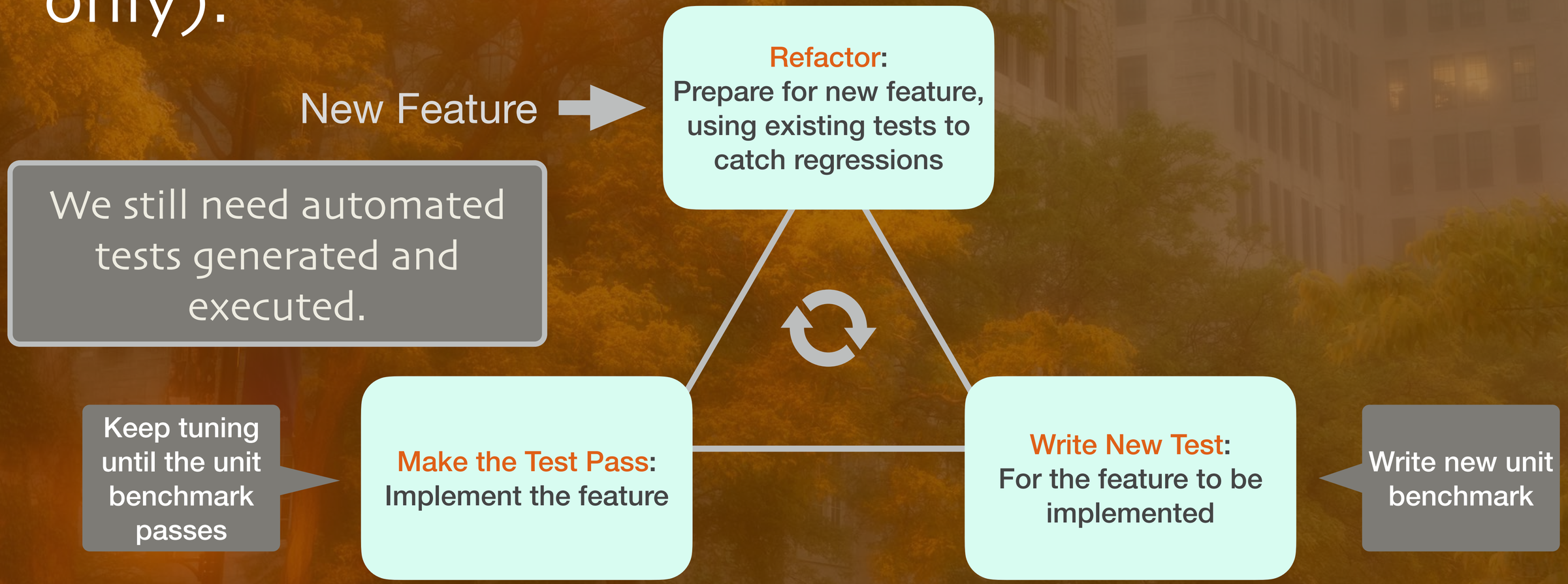
Maybe we have to abandon the idea of deterministic testing, but approach the challenge in a new way.

From Testing to Continuous Tuning

- What if we switch from testing for desired behavior to tuning for desired behavior?
  - We already tune models to improve domain-specific knowledge, chatbot behavior, etc.
    - Today: it's only done during model creation.
    - Tomorrow: continuously tune incrementally.

From Testing to Continuous Tuning

- Changes to the TDD cycle (for model behaviors only):



Thank you!

thealliance.ai
dwampler@thealliance.ai
Mastodon and Bluesky: @deanwampler

aialliance.org

deanwampler.com/talks