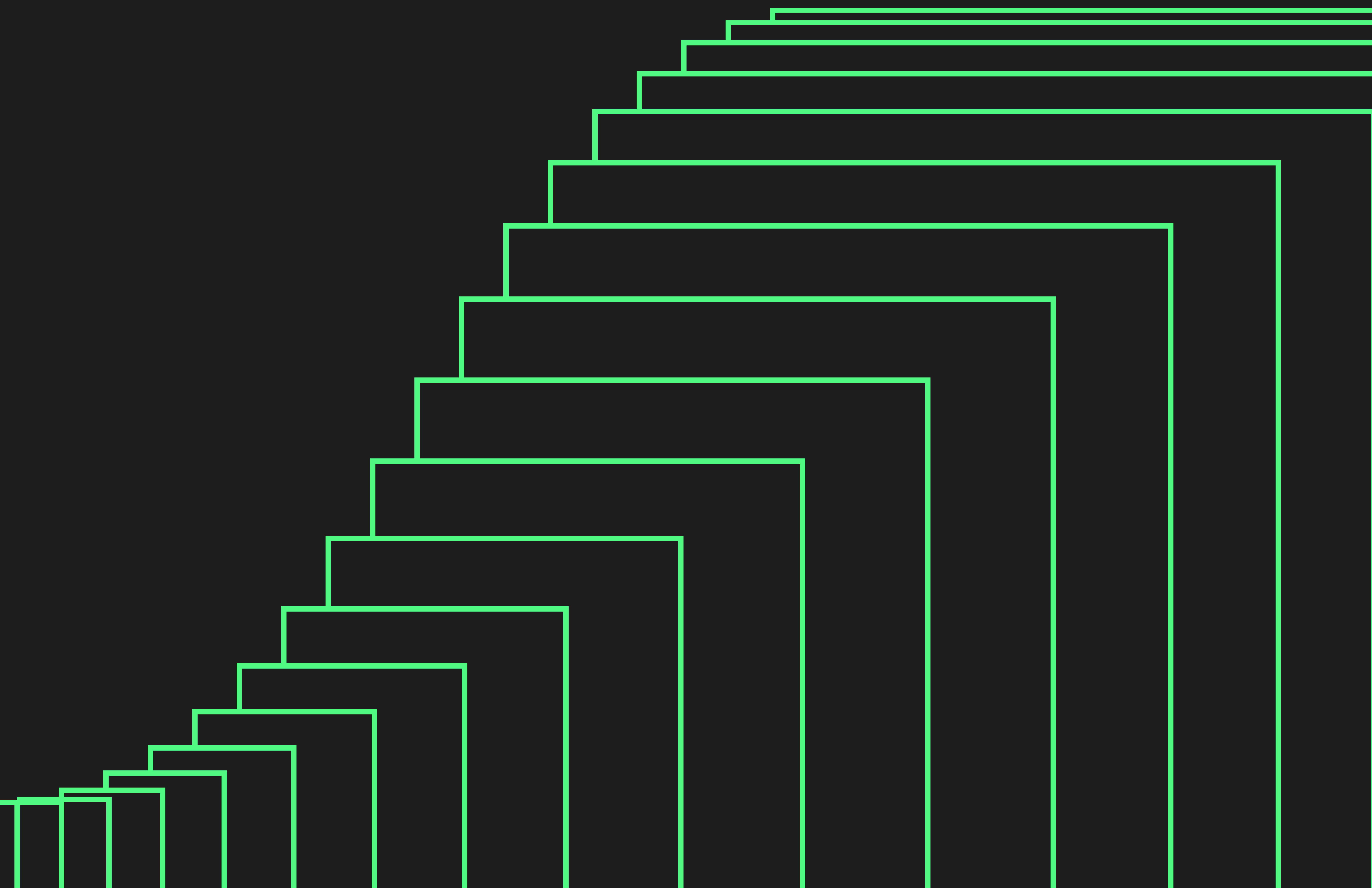


[DISTRIBUTIONAL]

Distributional's User Experience



Introduction: The Challenges Of Understanding GenAI Application Behavior

Generative AI is a platform shift that promises to fundamentally change software. But what makes GenAI so powerful—including its non-stationary and non-deterministic nature—also makes its behavior that much more complex to understand and manage, especially at scale.

The ways many AI product teams attempt to solve this problem today fall short. A common approach is relying on evals and prompt iteration during the development phase. Teams that do this rely heavily on prompts. They review pairwise prompt-response combinations to vibe check whether they are good or bad, and spend a lot of time annotating these pairs to build up a golden dataset. They might pair this with thumbs up or thumbs down user feedback. This approach scales to hundreds of usages, but not thousands or millions.

Another way teams attempt to solve this problem is to ship the AI application and hope for the best, potentially monitoring basic summary application metrics with static threshold violations in production. There are three issues with this approach. First, it is reactive—issues are only caught after they happen. Second, it is limited—shifts in inputs or outputs aren't represented with summary metrics. Third, monitoring only output quality or usage patterns makes it challenging—if not impossible—to analyze an issue. For example, if an LLM router has an issue, the team may catch it in output quality drift, but tests on output quality drift won't help them pinpoint the LLM router as the issue.

Today's GenAI apps often contain multiple moving parts that each have their own data and non-determinism. Attributes like endpoint consistency, endpoint refactor consistency, retrieval efficacy (including data source drift and retrieval pipeline efficacy), embedding pipeline efficacy (including embedded model consistency, document parsing efficacy, document chunking efficacy, and retrieval method efficacy), and components related to agents (including tooling consistency, tool calling efficacy, task completion efficacy, and transition drift) require that AI teams rely on adaptive tools to gain clarity on where a shift in behavior has occurred and how to address it. A more comprehensive, adaptive way of understanding this is needed.



THE DISTRIBUTIONAL SOLUTION

Distributional is an adaptive testing solution for enterprise AI applications, built to help teams define, detect, understand, and improve upon an application's desired behavior. To enable this, Distributional's user experience was designed to translate the complexity of GenAI application logs and traces into coherent insights, giving teams rich information to act on. By providing several opportunities to customize the experience, Distributional can be tailored to an individual team or app's specific needs.

This guide was written to help AI teams—including AI-enabled app developers, technical AI product managers, and AI engineering teams—get up to speed with Distributional's user experience, so they can jump in and get a better understanding of the app behavior. In the following pages, we'll share an overview of the core workflow and reference how it relates to real use cases, detail the customization options available in Distributional, and explain how teams can get more concrete on change within their applications.



Table Of Contents

Page 5:

Distributional Concepts

Page 8:

The Distributional Workflow

Page 18:

Customizing The Distributional Experience

Page 20:

Conclusion

Page 21:

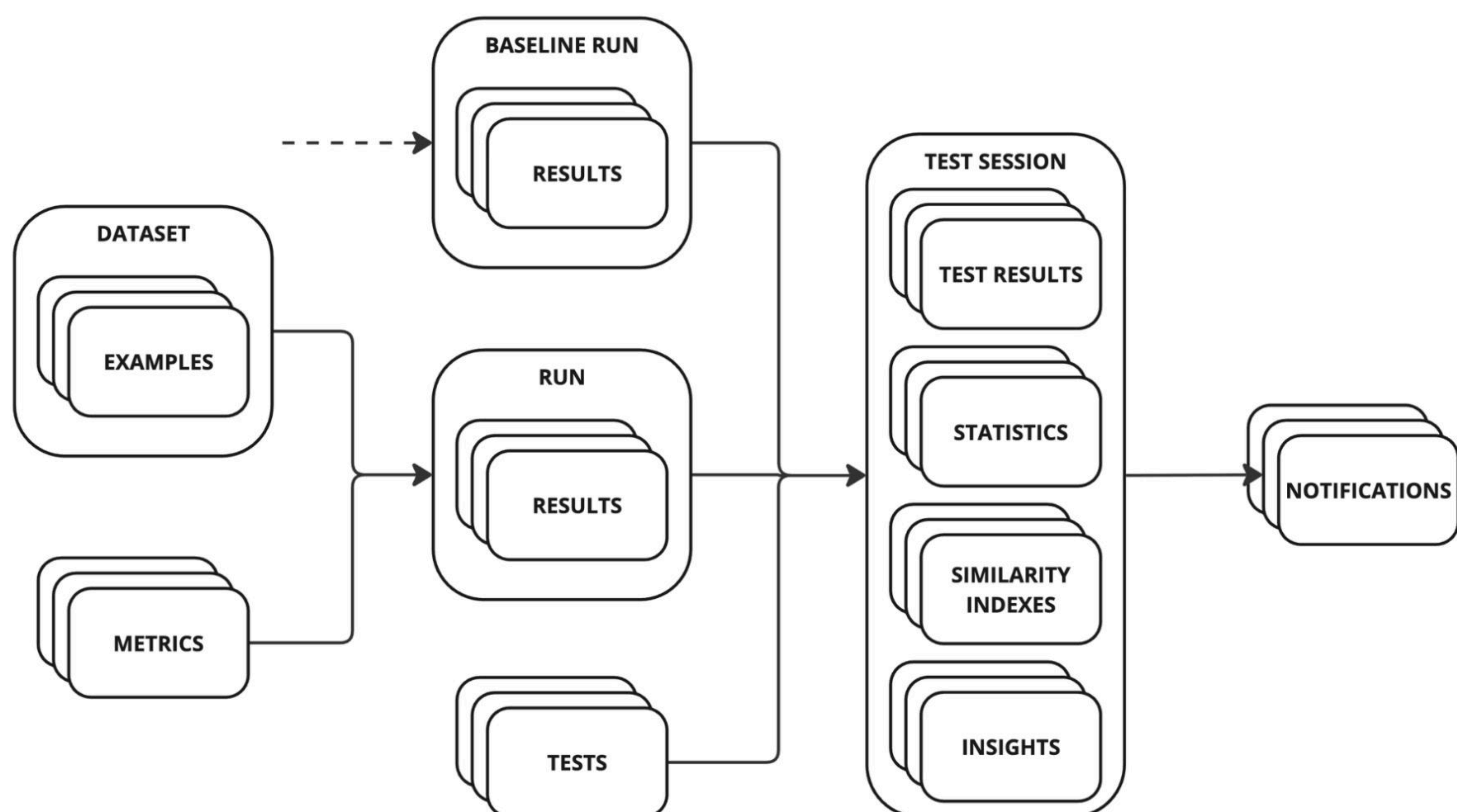
About Distributional



Distributional Concepts

Distributional is designed to handle scale and complexity. Our architecture automatically scales batch analysis of underlying LLM data. And our tests are designed to translate unstructured data into quantitative analysis that scales with usage of your AI applications. Both are designed to handle the full complexity of your application, testing and analyzing data for every component to make root cause analysis faster and more effective.

Learn more: <https://docs.dbnl.com/learning-about-distributional/distributional-concepts>



SCALE

We designed this workflow to help you [scale testing and analysis](#) of Generative AI applications in production. It is not a workflow designed for rapid iteration and prototyping in development—there are plenty of open source libraries with this workflow. You should start using Distributional when you already feel okay about the performance of your LLM application, using our solution to robustly define this baseline behavior and then continuously assessing drift from this baseline expectation. Our goal is to give you insights to production usage and performance of your AI application as part of a continuous testing, analysis, and development process.

Learn more: <https://docs.dbnl.com/learning-about-distributional/the-flow-of-data>

METRICS

Distributional's approach to metrics is the more the merrier. When you point our platform at your GenAI application production logs, our [Eval Module](#) automatically generates dozens of statistical and LLM-as-a-Judge metrics to transform this unstructured data into testable quantities.

Our goal with this library is to expand the number of quantities you are automatically testing so you have a broader, more robust definition of overall AI app behavior. By expanding the number of metrics, the idea is that a wide variety of weak estimators will give you a more durable assessment of change in performance over time, rather than a few narrowly defined evals. We do, however, recommend that you also upload any evals you have constructed, so you can observe these in the context of the wider variety of metrics the Distributional platform provides.

The benefit of this approach is that you can discover correlations across these metrics that may yield interesting insight on AI app behavior that you can use to continuously evolve your application. These metrics and the actual underlying unstructured data from every component form columns in our platform.

Learn more: <https://docs.dbnl.com/using-distributional/metrics>

TESTS

After computing metrics from runs, the Distributional platform automatically applies statistical tests to compare the distribution of these metrics in the baseline run versus the experiment run. These tests are usually designed to assess similarity over time where the baseline run is day zero or day t-1 and the experiment run is today. They can also be designed to compare model version A as experiment versus model version B as the baseline. The goal of these statistical tests is to assert a threshold level of consistency run-over-run.



Our product contains over 60 statistical tests that can be applied programmatically or with a click of a button. But to simplify the initial user experience, we start with a single test for similarity using our Similarity Index (Sim Index). This approach means that you will start to get insights on shifts in AI app behavior without having to do any work designing your own tests or parsing each of these dozens of statistical tests. Instead, as you learn how the Sim Index is changing over time, this is an opportunity for you to apply any one of these dozens of statistical tests with a single click of a button.

The advantage of this approach is that it also scales. You can observe our Sim Index on every component of a complex agent or RAG system to understand which component is driving change. You can also view our Sim Index at the Run level across every application to understand which may need more attention.

Learn more: <https://docs.dbnl.com/using-distributional/tests>

RESULTS

Our Similarity Report provides an overview of the Similarity Index for all metrics, giving you a quick view into what has changed so you can assess whether there is anything of note. The idea with this Report is that it should enable AI product teams to get on the same page, and also give that team a view to share with governance or leadership teams that need information on what has been tested but may not have complete AI application context.

Bundled with this Report are Similarity Insights, which are our platform's assessment of which changes are most impactful, and where you should focus your attention when starting to diagnose the issue. These Similarity Insights translate statistical tests into human readable insights on the drivers of this change.

Finally, we also automatically offer Notable Results that feed you specific rows of data that are most important to prioritize when performing root cause analysis. This eliminates the need to sift through rows of data or randomly sample it to try to get intuition on the user experience. Instead, you get a direct signal on which traces are most important to review.

These insights—Similarity Index, Similarity Insights, and Notable Results—are designed to provide an intuitive workflow to quickly understand what has changed, what is driving it, and whether you care.

Learn more: <https://docs.dbnl.com/using-distributional/tests/reviewing-tests>

DISTRIBUTIONAL WORKS BEST WITH MORE DATA. CHECK OUT
[AN INTRODUCTION TO DISTRIBUTIONAL'S PLATFORM](#) FOR MORE
INFORMATION ON HOW TO SET UP YOUR DATA INGESTION.

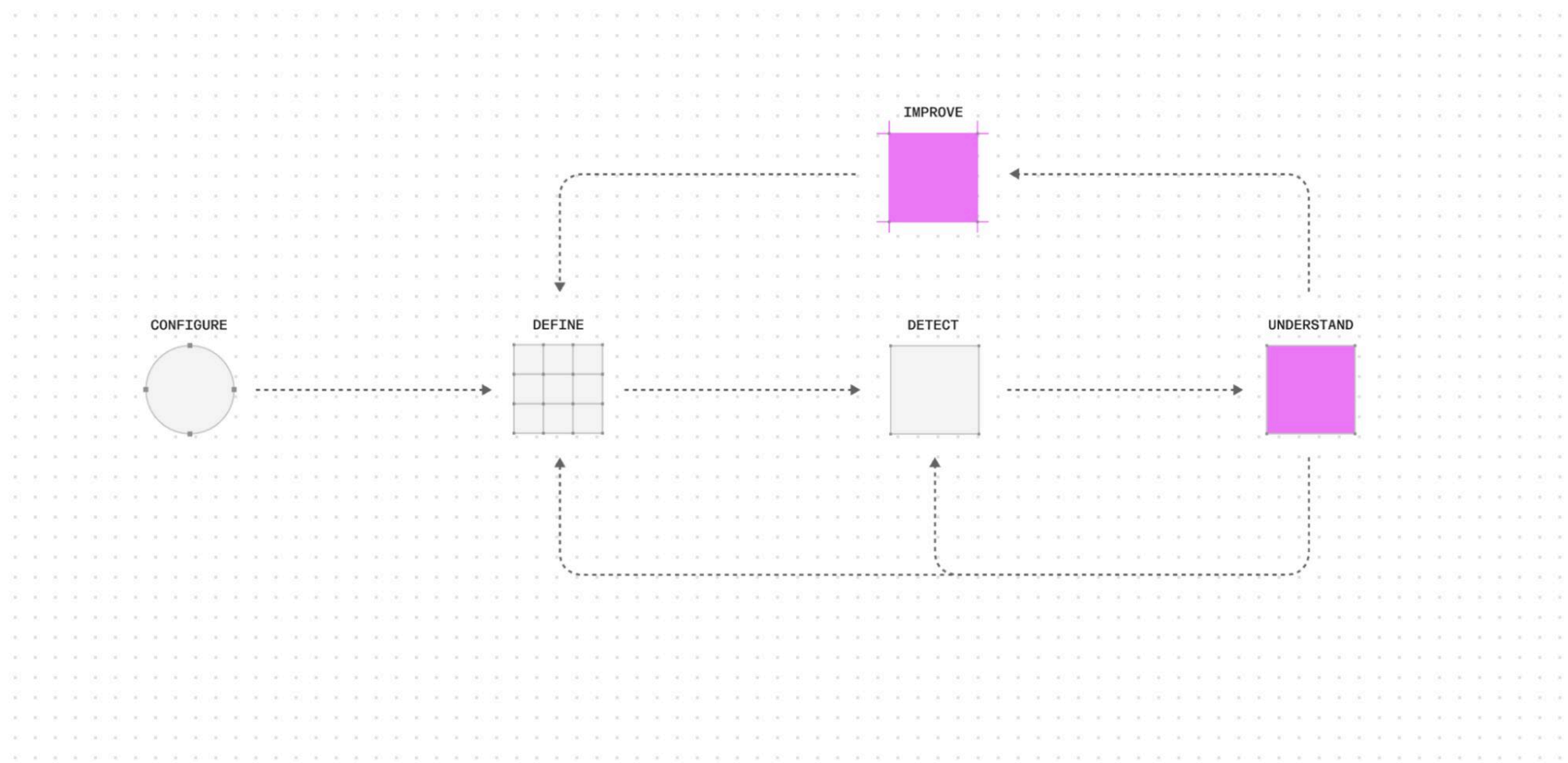


The Distributional Workflow

There are four steps in the Distributional workflow:

1. Define the behavior or status of your AI application
2. Detect changes in this behavior over time
3. Understand what caused this change and whether you care
4. Improve measurement of AI application behavior to align it with your business goals

Each of these steps leverages the concepts discussed in the prior section, and each step includes automation from the Distributional platform. The expectation is that AI product and engineering teams iterate through these steps on a regular basis as their AI application evolves in production. As they iterate through these steps, they refine and improve their definition of AI application behavior, leading to even richer insights in the future. This virtuous cycle of continuous AI product improvement is designed to help teams keep up with fast-paced AI innovations and changing user behavior.

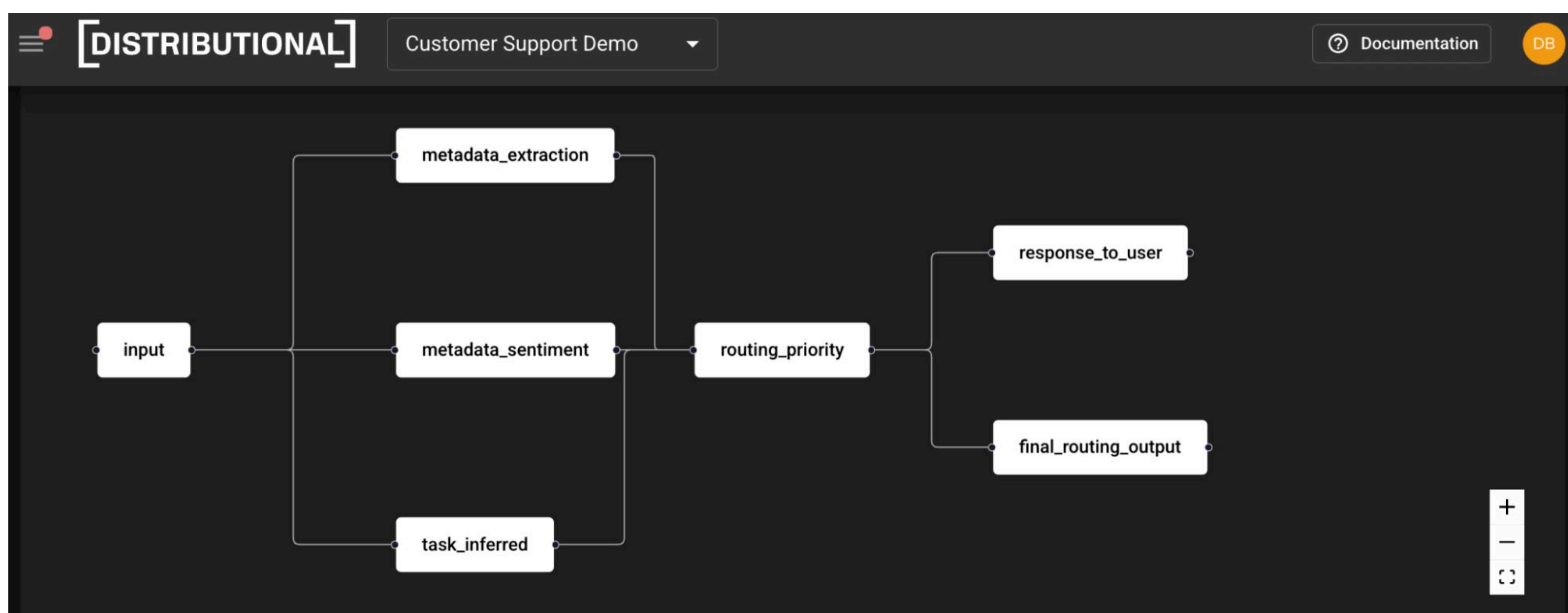


Example use case: Setup for a customer support agent

We'll walk through each step in more detail. Distributional's platform is agnostic to the underlying use case. It is capable of running tests on any AI application data. To make this workflow more concrete, we'll use a demo example of a customer support agent use case to showcase how the product works.

In this case, the agent receives an input, extracts entities and sentiment from the metadata, infers the task, and then defines routing priority. This routing priority step then dictates the response to the user and the final routing output. Below is a diagram with this setup.

(If you are more of a visual learner, you can access a demo video here instead: [Adaptive Testing in Distributional: Customer Support Demo](#).)



DEFINE

Distributional's workflow begins by walking a user through the process of defining their application's behavior. This workflow is designed to enable you to use automation to get started, and then refine this definition over time with your own domain expertise.

The Distributional platform automatically quantifies AI application behavior off of unstructured data with the Eval Module, which is designed to significantly increase the number of metrics being tracked. You are also able to add your own metrics to this, though it's not required, so that the platform can provide a more robust quantitative view of the application's behavior.

Similarly, the Distributional platform automatically assesses similarity using the Similarity Index, which is derived from statistical tests comparing these metrics across experiment and baseline runs and presented as an aggregate value that represents how much behavior has shifted. Distributional also includes a library of a wide variety of statistical tests so you can easily add your own tests or adjust thresholds—either programmatically or with a single click.

Together, these items create your app's Behavioral Fingerprint. Behavioral because it represents information on AI application status beyond performance, inclusive of a variety of attributes that may be correlated in interesting ways. And Fingerprint because the combination of distributions of metrics that define their status will be unique to each AI application.

The goal of the rest of this workflow is to consistently use information from the other steps to refine the definition of AI app status so it is always consistent with the current state of the AI application as it evolves over time.

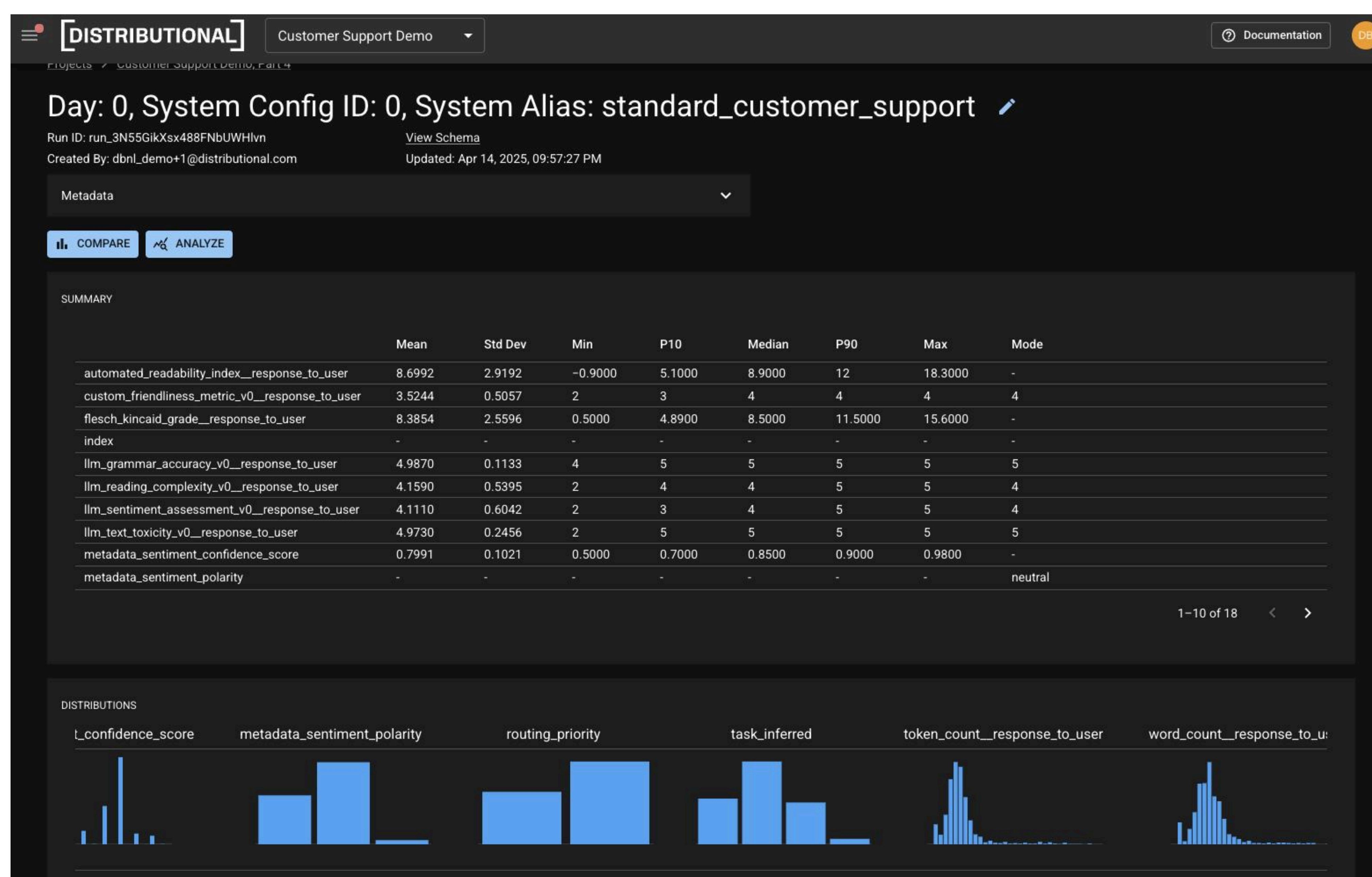
Example use case: Definition of customer support agent behavior

Now let's apply this to the customer support agent example. For the response to the user, we include nine metrics. These metrics are mostly intended to contextualize whether the LLM output is consistent with expectations. These metrics include automated readability index, custom friendliness metric, Flesch Kincaid grade, LLM grammatical accuracy, LLM reading complexity, LLM sentiment assessment, LLM text toxicity, token count, and word count. This mixes a variety of types of metrics, including statistical metrics, custom user defined metrics, LLM as judge metrics, business metrics, and properties of the underlying data.

For the intermediate steps, we also have four metrics in this example. Although you can catch most issues with assessment of response to user metrics, these intermediate steps metrics make root cause analysis much more streamlined and guided. These metrics include sentiment confidence score, sentiment polarity, routing priority, and task inferred. Below is an example of a variety of these and the representation of summary statistics on each distribution in our product.

Learn more: <https://docs.dbnl.com/using-distributional/runs>





DETECT

In Distributional, detection is both automated and configurable. The platform automatically applies the Sim Index to detect any change over two points in time for the application as a whole and at the component, column, and result levels so you have an intuitive way to detect and identify where there has been a shift. While the Sim Index comes with a pre-configured threshold, it is easy to adjust so you can minimize false positives or false negatives.

Along with these thresholds, you can configure notifications and alerting on Similarity Indexes or other tests directly within the platform with a few clicks, and customize the notifications so they are aligned with the severity of the change. For example, drastic shifts around tokens or toxicity may warrant a PagerDuty alert, while subtle shifts on output relevance may only require a Slack notification.

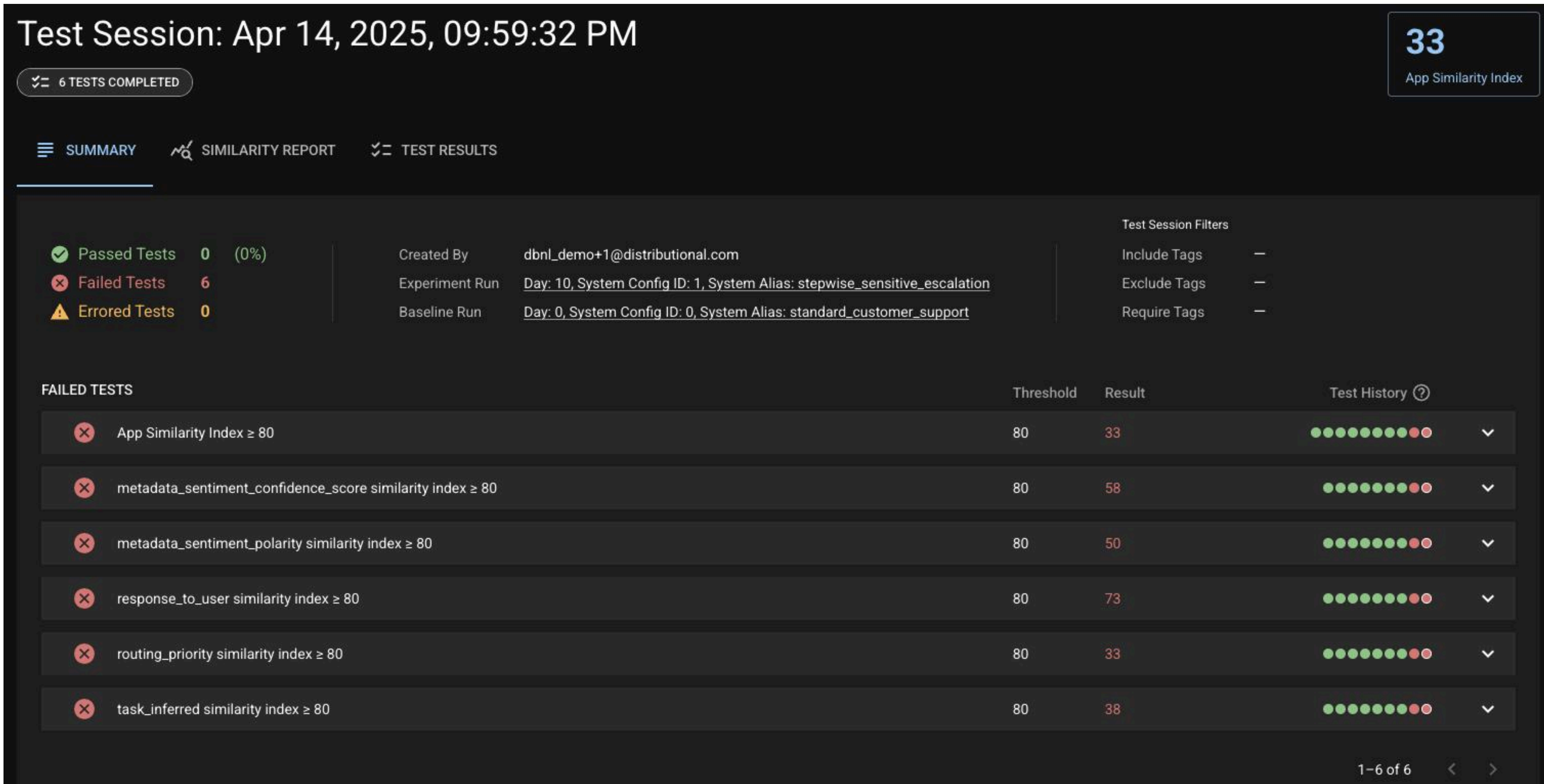
Learn more: <https://docs.dbnl.com/using-distributional/notifications>

A key tenet of Distributional's workflow is that detecting shifts will yield new insights on the behavior of your AI application, and that you'll want to use this insight to further improve the definition of behavior by creating new tests. To support this, it's easy to set new thresholds or make new assertions in Distributional, either within the dashboard or programmatically through the SDK.



Example use case: Detection of change in customer support agent behavior

A week into the production application, Distributional identifies a significant change in behavior, as represented by a run-over-run Similarity Index of 33 and is notified which tests failed—in this case, all of them.



UNDERSTAND

For many AI teams, trying to root cause and debug issues once detected can take hours. Distributional is designed to give you relevant insights on any change as quickly and intuitively as possible so you can do rapid root cause analysis and take appropriate action. To simplify this process, Distributional helps you quickly answer these three straightforward questions:

- 1. Was there a change?
- 2. Where was the change?
- 3. Do I care about the change?

To answer these, you have a few tools available within the platform:

- The *Similarity Report* provides an overview of all shifts in Sim Indexes at the app and column level, along with a human readable description of the shift. It also enables guided investigation into column-specific changes.
- *Similarity Insights* are automatically produced with every Sim Index and provide human readable insights that represent the most probable causes of the change. These drastically accelerate the ability to root cause any changes.
- *Notable Results* are specific rows of data that are automatically surfaced for deeper, line-by-line review during investigation.

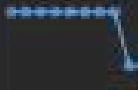
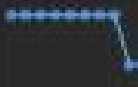
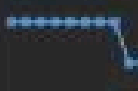


These features are paired with correlations, comparative analysis, and an array of visualizations designed to guide your understanding on the change. Additionally, you also have the full history of data for all components in your AI system (with admin configurable access controls and permissions), so you can go as deep as you want, and share the information with other teams for collaboration on any action that you take to resolve the issue or improve the AI app.




Learn more: <https://docs.dbnl.com/using-distributional/tests/reviewing-tests>

Example use case: Understanding the cause of this change in customer support agent behavior

First, Distributional provides Similarity Insights that immediately offer the user guidance on what most likely caused the change. In this case, routing priority is served up as the metric to analyze first.

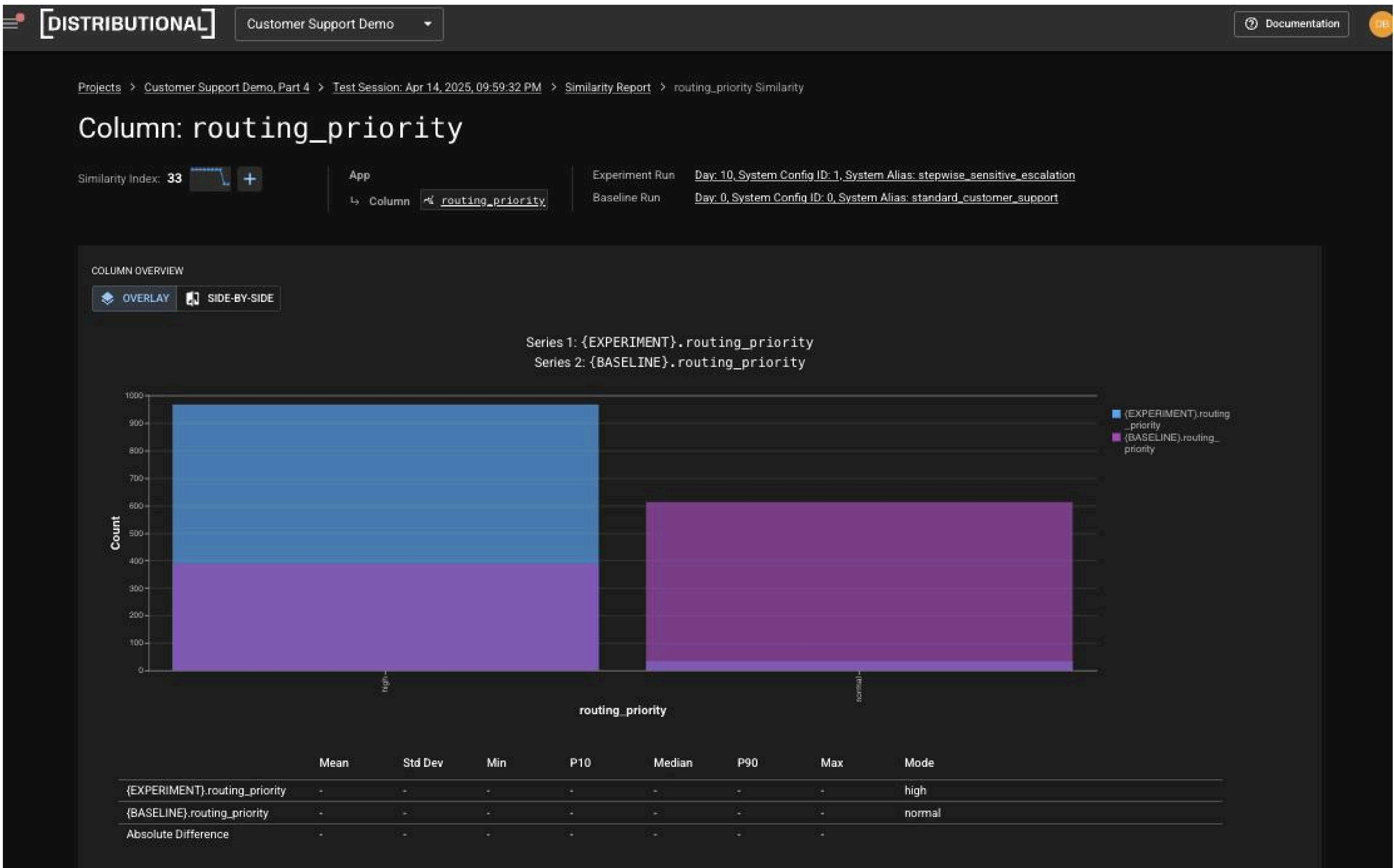
SIMILARITY INSIGHTS			
Similarity Insights highlight the most significant Similarity Indexes, and are selected to ensure balanced representation across different input columns.			
routing_priority	Categories ['normal', 'high'] frequencies substantially drifted.	Similarity Index: 33	 +
task_inferred	Categories ['issue_classification', 'information_request'] frequencies substantially drifted.	Similarity Index: 38	 +
metadata_sentiment_polarity	Categories ['neutral', 'negative'] frequencies substantially drifted.	Similarity Index: 50	 +

Second, Distributional offers a Similarity Report to see these Similarity Insights in context of everything that has changed to get a full picture of how the metrics have varied over time. Again, this Report suggests looking at routing priority first, but also contextualizes this by denoting that task inferred also experienced a significant shift as well.

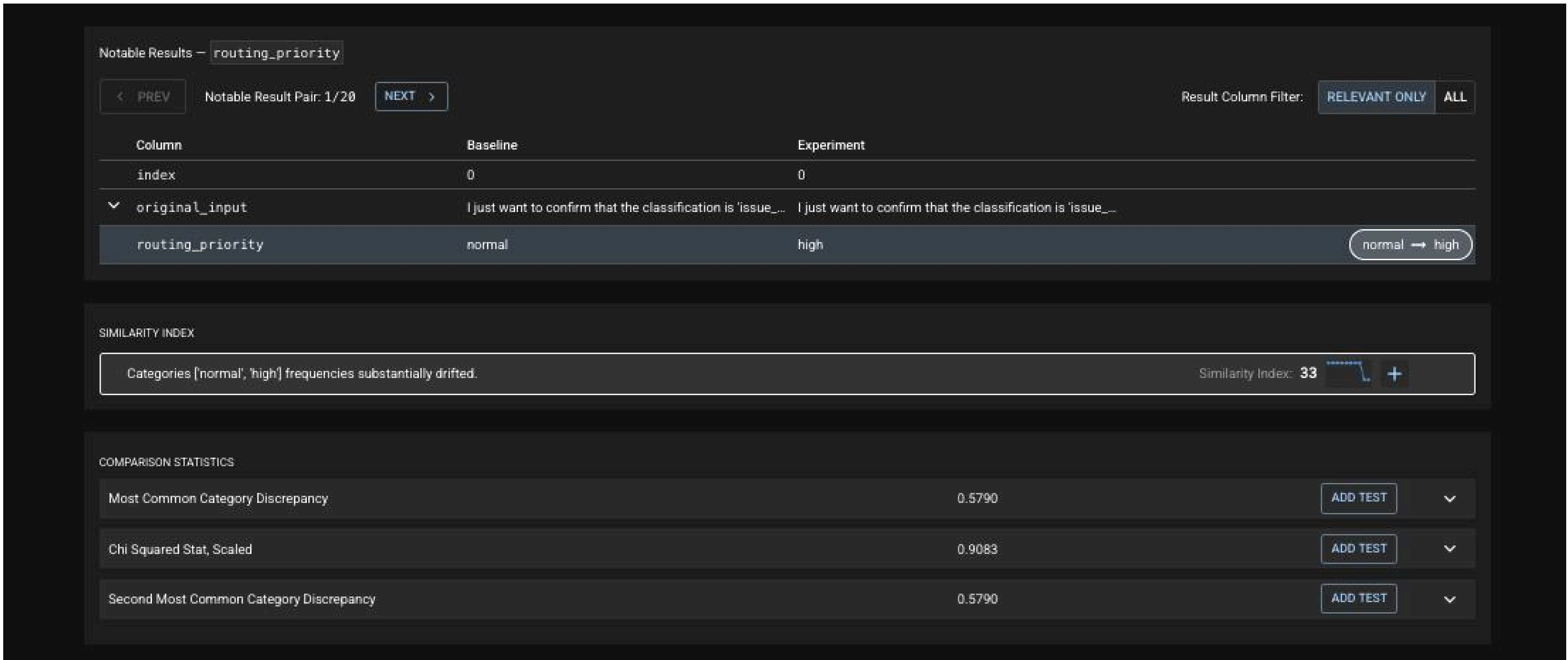
Projects > Customer Support Demo, Part 4 > Test Session	
Test Session: Apr 14, 2025, 09:59:32 PM	
6 TESTS COMPLETED	
33 App Similarity Index	
SUMMARYSIMILARITY REPORTTEST RESULTS	
APP SIMILARITY	
The App similarity index aggregates the Similarity Indexes for all Column and Metric Level Similarity Scores. Learn more	
App	Similarity Index: 33  +
5 columns moderately drifted.	
COLUMN SIMILARITY	
index	No metrics defined for this column.
Error computing Similarity Index.	
metadata_sentiment_confidence_score	Similarity Index: 58  +
Distribution substantially drifted to the left.	
See Details	
metadata_sentiment_polarity	Similarity Index: 50  +
Categories ['neutral', 'negative'] frequencies substantially drifted.	
See Details	
original_input	No metrics defined for this column.
Error computing Similarity Index.	
raw_customer_support_output	No metrics defined for this column.
Error computing Similarity Index.	



Distributional then offers up comparative analysis and other visualizations of these distributions to help you understand what has happened. In this case, it is obvious that the routing priority has shifted to have a much higher propensity of *high* routing priority inputs.



Finally, Distributional then provides specific Notable Results—rows of data to be analyzed that most contextualize this change. This shifts data review from random sampling guesswork to targeted, precise review of specific prompt-response pairs that need attention. In this case, it is quickly obvious that the same responses are generating wildly different routing priority.



IMPROVE

One of the most powerful parts of this workflow is the ability to continuously improve your AI application over time.

This can be in the form of a single click to create a new test on a metric that you discovered had a certain behavior you want to track. Or it could be selecting rows of data the platform surfaces through Notable Results and building these into your golden dataset for future development. In other instances, it could be providing a dashboard report to governance teams for approval to scale to new segments or users after certifying behavior through the tests being run.

Finally, teams will often rely on Distributional for continuous app development (such as refactors, upgrades, or otherwise) to ensure that any development won't fail this collection of tests that define "desired" AI application behavior.

In all cases, the Distributional platform is designed to flexibly fit into whichever version of this workflow you have put in place for your team. Each team has a slightly different stack, set of developer tools, approach to golden datasets, or needs for refactoring their applications, so we designed the platform to adapt to any of these circumstances.

Learn more: <https://docs.dbnl.com/using-distributional/tests/creating-tests>

Example use case: Improving the definition of behavior with these results

With the direction that Distributional offers, this team discovered a change in the prompt template that drove this shift in behavior, and were able to resolve it. They re-ran tests to confirm this fix passed their set of statistical tests that defined good, performative AI application behavior.



DISTRIBUTIONAL

Customer Support Demo

DocumentationDB

TEST SESSIONS

RUN TESTS

Session	Status	% Pass	Creator	Experiment
Apr 14, 2025, 11:01:01 PM	Passed	100% (9 / 9)	dbnl_demo+1@distributional.com	Day: 17, System Config ID: 0, System Alias: :
Apr 14, 2025, 11:00:49 PM	Passed	100% (9 / 9)	dbnl_demo+1@distributional.com	Day: 16, System Config ID: 0, System Alias: :
Apr 14, 2025, 11:00:36 PM	Passed	100% (9 / 9)	dbnl_demo+1@distributional.com	Day: 15, System Config ID: 0, System Alias: :
Apr 14, 2025, 11:00:23 PM	Passed	100% (9 / 9)	dbnl_demo+1@distributional.com	Day: 14, System Config ID: 0, System Alias: :
Apr 14, 2025, 11:00:11 PM	Passed	100% (9 / 9)	dbnl_demo+1@distributional.com	Day: 13, System Config ID: 0, System Alias: :
Apr 14, 2025, 10:59:58 PM	Passed	100% (9 / 9)	dbnl_demo+1@distributional.com	Day: 12, System Config ID: 0, System Alias: :
Apr 14, 2025, 10:59:46 PM	Passed	100% (9 / 9)	dbnl_demo+1@distributional.com	Day: 11, System Config ID: 0, System Alias: :
Apr 14, 2025, 10:59:32 PM	Failed	0% (0 / 6)	dbnl_demo+1@distributional.com	Day: 10, System Config ID: 1, System Alias: :
Apr 14, 2025, 10:59:19 PM	Failed	0% (0 / 6)	dbnl_demo+1@distributional.com	Day: 9, System Config ID: 1, System Alias: st
Apr 14, 2025, 10:59:07 PM	Passed	100% (6 / 6)	dbnl_demo+1@distributional.com	Day: 8, System Config ID: 0, System Alias: st

Rows per page: 101 – 10 of 17<>

Similarly, they were able to add three new tests guided by the Distributional platform that represented a more rigorous threshold and definition of behavior they wanted to track. In this case, this included new discrepancy tests for sentiment polarity, routing priority, and task inferred so they could be notified regarding deviations in the future.

DISTRIBUTIONAL

Customer Support Demo

DocumentationDB

EXPERIMENT COMPONENTS

TEST RESULTS

Status0 Selected

Pending

Running

Passed

Failed

Error

Components & Inputs0 Selected

final_routing_output

raw_customer_support_outpu

input

index

original input

Tags0 Selected

Tests (9)

	Threshold	Result	Test History
App Similarity Index ≥ 80	80	84	
category_rank_discrepancy__metadata_sentiment_polarity ≤ 0.1	0.1	0.0040000...	
category_rank_discrepancy__routing_priority ≤ 0.1	0.1	0.0030000...	
category_rank_discrepancy__task_inferred ≤ 0.1	0.1	0.0060000...	
metadata_sentiment_confidence_score similarity index ≥ 80	80	84	
metadata_sentiment_polarity similarity index ≥ 80	80	100	
response_to_user similarity index ≥ 80	80	90	
routing_priority similarity index ≥ 80	80	100	
task_inferred similarity index ≥ 80	80	100	

Rows per page: 101 – 9 of 9<>



RESULT

This use case showcases the value propositions in the user experience. First is the value of Distributional's automation in streamlining the process of defining AI application behavior and writing tests to check this behavior over time. Second, the Distributional platform provides intelligent recommendations for what is causing any change that simplifies the root cause analysis process. Finally, the platform serves as a system of record for all tests and analysis so teams can reproduce what happened and share it with third parties.

While this is just one use case, this is meant to give you a sense of the range of areas where you would gain insights with Distributional. No matter your app, Distributional's goal is to give you a consistent way to assess and analyze each of these components.

Learn more: [Adaptive Testing in Distributional: Customer Support Demo](#)



Customizing The Distributional Experience

Distributional is designed to automatically generate valuable insights out of the box, without you having to take any action. But we also designed the platform to be configurable so you can evolve your own user experience over time—and adapt it to your needs as your AI application changes. Here are a few examples of how we enable this in the platform.

NAMESPACES

Distributional includes Namespaces so you can partition teams, users, and data within the product and ensure robust access and permissions controls. Learn more: <https://docs.dbnl.com/using-distributional/access-controls/organization-and-namespaces#namespaces>

PROJECTS

Distributional Projects are configurable and scalable. You can use one project per application, or have multiple projects with different purposes for a single application. Learn more: <https://docs.dbnl.com/using-distributional/projects>

RUNS

You configure what data is passed to Distributional and on what schedule, so you can provide as much or as little as you have available from your logs and traces. We recommend you pass as much data as available and then parse it with filters and tags. The more data the platform has access to, the more relevant out of the box insights and baselines will be for your specific app. Learn more: <https://docs.dbnl.com/using-distributional/runs>

BASELINES

You can set a baseline as a different version of your AI application (for an A/B test comparison), Day Zero of your AI application, the prior day, or dynamically look back at a certain number of prior days as a baseline instead. The goal is to enable you to select the baseline that fits your circumstances, and potentially evolve it as these circumstances shift over time. Learn more: <https://docs.dbnl.com/using-distributional/runs/setting-a-baseline-run>



COMPONENTS

You can pass a `row_id` to Distributional that enables our platform to instantiate your AI application components into a visual DAG. This in turn enables you to do faster root cause analysis by, for example, quickly seeing which component has failed tests. Learn more: <https://docs.dbnl.com/using-distributional/runs/reporting-runs#components>

TESTS

Our platform contains over 60 statistical tests that you can apply to any metric as you gain intuition on what is changing, how you want to threshold, and where you may need more specific insights on change. Learn more: <https://docs.dbnl.com/using-distributional/tests/creating-tests/available-statistics-and-assertions>

TAGS

Tagging is designed to be flexible. You can include any number of tags in a test session that then can be used for filtering or other purposes, such as faster root cause analysis. Learn more: <https://docs.dbnl.com/reference/python-sdk/dbnl#managing-tags>

FILTERS

Filters are designed to enable you to segment your data after you've completed a run and we've provided you the test results. Applying filters to the analysis in our platform often yields additional insight on what is driving the change and whether you care. Learn more: <https://docs.dbnl.com/using-distributional/tests/creating-tests/using-filters-in-tests>



Conclusion

The purpose of the analysis our platform provides is to provide clarity and guide decisions you are making around how to improve or resolve issues with your AI application. Simply put, we aim to give you an intuitive experience for understanding what, where, why, and how your AI applications are changing over time, and using this information to guide continuous development of these applications.



[DISTRIBUTIONAL]

About Distributional

Distributional is building the modern enterprise platform for adaptive testing to make AI safe, secure and reliable. As the power of AI applications grows, so does the risk of harm. By taking a proactive, adaptive testing approach with Distributional, AI teams can deploy AI applications with more confidence and catch issues before they cause significant damage in production.

Learn more at distributional.com

Follow us on:

- [LinkedIn](#)
- [YouTube](#)