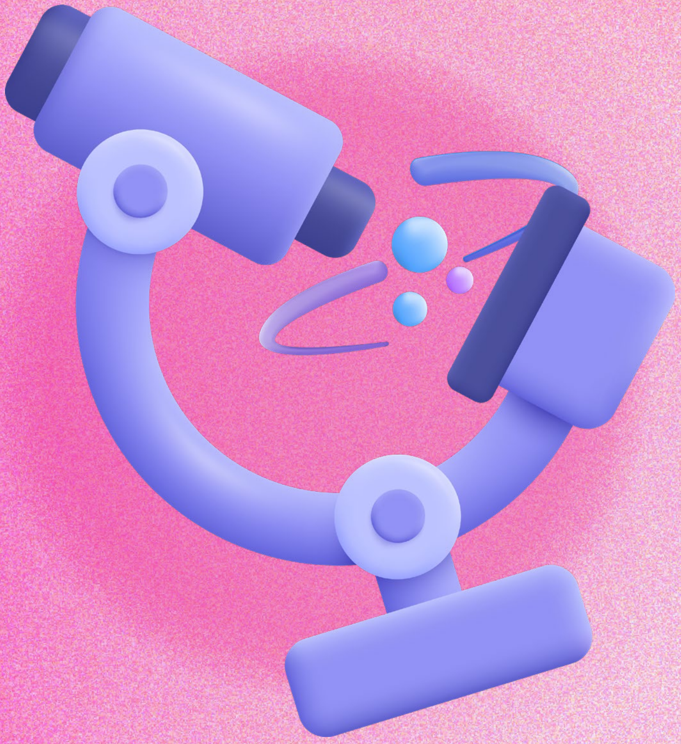
arize

THE DEFINITIVE

LLM Observability Checklist for Healthcare, Life Sciences and Consumer Health

What to look for when assessing large language model (LLM) observability platforms



Introduction

Healthcare, life sciences and consumer health are undergoing a quiet revolution in generative AI. Early applications are showing promise on everything from speeding up preclinical drug discovery and development by better predicting molecular behavior to augmenting how physicians and providers provide care to their patients.

Given the potential harms and regulatory risks intrinsic to applying AI in healthcare, having robust LLM evaluation and LLM observability is critical.

How can teams deploy generative AI reliably and responsibly – and what should they look for when assessing partners? Informed by experience working with top researchers and providers that have successful LLM apps deployed in the real world, this checklist covers essential elements to consider when assessing an LLM observability provider.

Example Use Cases

Here are several use cases for LLMs in healthcare, with associated LLM observability needs highlighted (full descriptions of each, such as "Traces and Spans," appear in more detail in the sections below).

Summarization

LLM Use Case

Specialized summarization apps can be useful in helping providers navigate a deluge of new research or prior notes.

Key LLM Observability Capabilities

- Traces and Spans
- LLM System Evaluation
- Retrieval Augmented Generation
- Embeddings

Treatment Recommendations

LLM Use Case

Speed time-to-diagnosis with LLM copilot or chatbot apps that summarize patient visits and leverage electronic medical records and validated information to provide treatment recommendations.

Key LLM Observability Capabilities

- Traces and Spans
- LLM System Evaluation

Chat-To-Diagnosis

LLM Use Case

Leveraging medically validated information grounded in the latest available clinical guidelines to arrive at a potential diagnosis.

Key LLM Observability Capabilities

- Traces and Spans
- LLM System Evaluation
- Retrieval Augmented Generation
- Embeddings

Example Use Cases

Copilot

LLM Use Case

Useful in areas like checking on prescriptions to avoid bad drug interactions.

Key LLM Observability Capabilities

- Traces and Spans
- LLM System Evaluation

LLM System Evaluations

Often, the first step in troubleshooting an LLM-powered task or application is identifying where it responded poorly or hallucinated. This is a nontrivial problem for several reasons: user feedback or any other “source of truth” is often extremely limited, human labeling is expensive, and it is easy to make LLM applications unnecessarily complex given several lines of code can trigger dozens of calls. Here are several essentials for a platform helping with LLM-assisted evaluation.

□ Modifiable and Runs Across Environments Seamlessly

There are constantly new system evals and templates pre-tested on a lot of datasets being released. In practice, what actually matters is having the flexibility to modify templates and run across environments seamlessly. Evals should be able to run on dataframes, in Python pipelines, or in LlamaIndex or LangChain callbacks. They should also be applicable across your development lifecycle, because it’s vital to determine the optimal evaluation framework for your LLM-powered apps before deployment.

□ Custom Evals

In the event you are not adjusting an existing eval template, AI engineers should also be able to build a custom eval template to then run on a golden dataset and benchmark performance.

□ Fast Throughput

When building an app, it’s critical to run as fast as possible on batches of eval data and maximize the throughput and usage of your API key. Current call-by-call-based approaches prove insufficient for many enterprise LLM apps.

□ Support for Common Pre-Tested Templates

Proven, rigorously pre-tested eval templates and convenience functions for a set of common eval “tasks” – including code generation, RAG relevance, hallucinations, Q&A on retrieved data, toxicity, and summarization – is essential. Quality templates provide a quick, out-of-the-box way to build evaluation benchmarks for your app.

□

Metrics Beyond Average Accuracy

When picking an LLM evaluation framework, it's important to focus on precision and recall because they will tell you where to actually improve. Just knowing accuracy or the percent an LLM gets correct is insufficient to ensure an LLM application accomplishes a business-specific goal (i.e. minimizing costly false negatives in medical testing).

□

Ability To Run Evals on the Span and Chain Level

In today's state of LLM development, a few short lines of code can kick off dozens of calls, creating a black box for troubleshooting problems that emerge. Span and chain level granularity of evaluation are key in surfacing and getting to the bottom of potential issues.

□

Reproducible

Evals should apply data science rigor to the testing of model and template combinations, transparently providing benchmark datasets and tests to reproduce achieved results for the eval task.

LLM Traces and Spans

Traces are a well-established concept in infrastructure observability, emerging more recently for AI systems as foundational models became commonplace. LLM-powered applications leveraging agents are often composed of a complex series of steps (spans) to achieve a higher-order objective. Spans might be non-ML software-defined tools (like a calculator) or other ML systems. LLM traces helps by building understanding of the system from the outside, troubleshooting and pinpointing novel problems (i.e. “unknown unknowns”).

□ Specification for Capturing and Storing All Relevant LLM Application Executions

The category of telemetry data that is used to understand the execution of LLMs and the surrounding application context such as retrieval from vector stores and the usage of external tools such as search engines or APIs is critical.

□ Support for LLM Traces

A trace records the paths taken by requests (made by an application or end-user) as they propagate through multiple steps. Without tracing, it is challenging to pinpoint the cause of performance problems in a system. A trace is made of one or more spans.

□ Comprehensive Ability To Log Span Kinds and Attributes

Capturing span kinds – chain, retriever, LLM, embedding, agent, embedding, reranker, or tool – is table stakes. Attributes are key-value pairs that contain metadata that you can use to annotate a span to carry information about the operation it is tracking. It’s important to select an observability provider that provides comprehensive support for your current and future use cases.

SECTION 03

Prompt Engineering

Prompt engineering is often the fastest and highest-leverage way to improve the performance of your LLM-based application. Frequently, LLM performance can be improved simply by swapping prompt templates, or iterating on the one you have.

□ Run On Any Major LLM Provider

An LLM observability platform should be agnostic and support the prevailing foundation models. This is also important to help you avoid model lock-in should you or your organization ever decide to switch or use multiple LLM providers.

□ Support for Prompt Variables

The ability to switch things out and run experiments across context and prompt templates is key to prompt engineering. Users should be able to: identify responses with poor feedback or evaluation scores and identify the template associated with poor responses.

□ Performance Analysis Across Templates

Ability to edit the prompt variables, LLM parameters, or the prompt templates and then re-run to easily compare responses for analysis prior to implementing a new or revised prompt template.

□ Compare Across Prompts and Templates

Instead of iterating on a single prompt, you should be able to see the impact across a number of datapoints. Instead of just running one prompt at a time, running across multiple prompts to see the impact of the changes on more data can speed up improvements.

Retrieval Augmented Generation (RAG)

In connecting an LLM to private or proprietary data, retrieval augmented generation is emerging as a powerful use case. Key functionality can help teams maximize results.

□ Support for RAG Metrics

The typical flow of retrieval is that a user query is embedded and used to search a vector store for chunks of relevant data, which might be semantically similar but not usable to answer the question. Leveraging an LLM-assisted eval can help analyze the performance of each chunk.

From there, traditional search and retrieval metrics – MRR, Precision @ K, and NDCG – help in determining how well a search and retrieval system is returning the right documents to a given context window.

□ Embeddings and UMAP Cluster Analysis

These metrics can be used by cluster (UMAP), making them very powerful to track down problems from the simplest to the most complex. Breaking up embeddings into groups of inferences using a HDBSCAN can be useful if you are trying to identify areas of your embeddings that are drifting or performing badly. UMAP can also help in understanding how embeddings have encoded semantic meaning in a visually understandable way.

□ Purpose-Built Workflows for Optimizing and Improving

During the building phase, robust testing and evaluation to inform chunking strategy and retrieval performance is important. Once in production, tools should help in determining whether to expand a knowledge base, refine chunking strategy or enhance context understanding.

Fine-Tuning

While a less-emphasized strategy given the high effort involved, fine-tuning essentially generates a new model that is more aligned with your exact usage conditions.

□ Workflows for Finding Priority Areas for Fine-Tuning

Leveraging user feedback as well as LLM-assisted evaluation and task-based metrics (i.e. ROUGE for summarization), a tool should assist in finding cohorts of bad or sub-optimal evaluation scores to speed workflows for improvement.

□ Data Export Controls

Ability to export sets of data or insights to work in another environment, such as a data science notebook, will help facilitate collaboration and further investigation with your main stakeholders.

SECTION 06

Embeddings Analysis

An embeddings troubleshooting view is important in helping to identify areas of drift and performance degradation.

□ Automatically Cluster Data for Anomaly Detection

Troubleshooting embeddings is incredibly hard. Teams require the ability to find groups of problems as clusters, and analyze those clusters relative to performance metrics or drift. The ability to automatically find clusters of problems and integrate workflows for fixing the data is required by any team using complex models.

□ 2-D and 3-D Embeddings Projector

Visualizing your data in a low dimensional space is critical to the process of finding the root cause of problems and analyzing them. Projecting embeddings to 2-D or 3-D space is imperative to understand the groups of data that might be causing issues, selecting them, filtering them and exporting them.

□ Sort Problematic Clusters by Performance Metrics

The ability to sort clusters of problems by performance metrics is critical to understanding what groups of clusters are causing problems and sorting what teams should look at first.

□ Interactively View Data, Select Data, Colorize Data and Export

The ability to interactively view data, select data, and colorize by performance/ features allows teams to work through interactive root cause workflows. Once you find the group of data causing the issue you can export or save the cluster of data for use in monitoring or fine tuning flows.

SECTION 07

General Platform Support

Selecting a technology partner that meets both current and future needs is critical.

□ Full-suite support for ML observability beyond LLMs

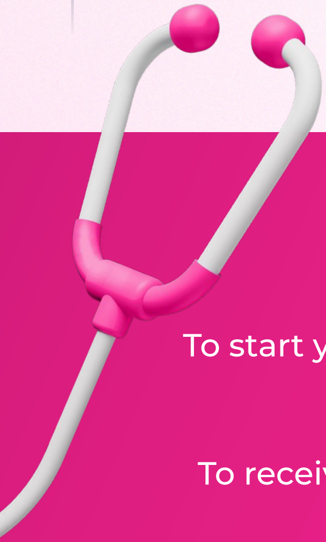
Your observability provider should support your full machine learning portfolio, not just LLMs. Even if your organization doesn’t currently deploy other ML models – such as for classification, computer vision, or ranking – in tandem with LLMs, the coverage is important to ensure you don’t paint yourself into a corner from a strategic perspective. If you currently deploy other ML model types anywhere in your organization, this criteria is a no brainer.

□ Self-Servability

In order to be developer and AI engineer friendly, it is essential for an LLM observability platform to offer easy-to-understand docs covering a wide range of prevailing use cases. Self-serve and open source options that enable practitioners to try for themselves rather than talking to a sales team are table stakes.

□ Customer Support with ML and Data Science Expertise

Given the rapid pace of innovation and research in generative AI, it’s critical to have trusted partners with a proven track record in data science and LLMOps to serve as advisors.



arize

To start your LLM observability journey, **sign up for a free account** or **schedule a demo.**

To receive more educational content, **Sign up** for our bi-monthly newsletter “The Drift”