IAGENTS | LLMS | OPENAI | GEMINI | PHOENIX | TRUELENS

Evaluating and Monitoring LLM Agents: Tools, Metrics, and Best Practices

This blog includes the tools that you can use to monitor and assess the performance of the Agentic approach

Chinmay Bhalerao · Follow

Published in Towards AI · 9 min read · Nov 17, 2024

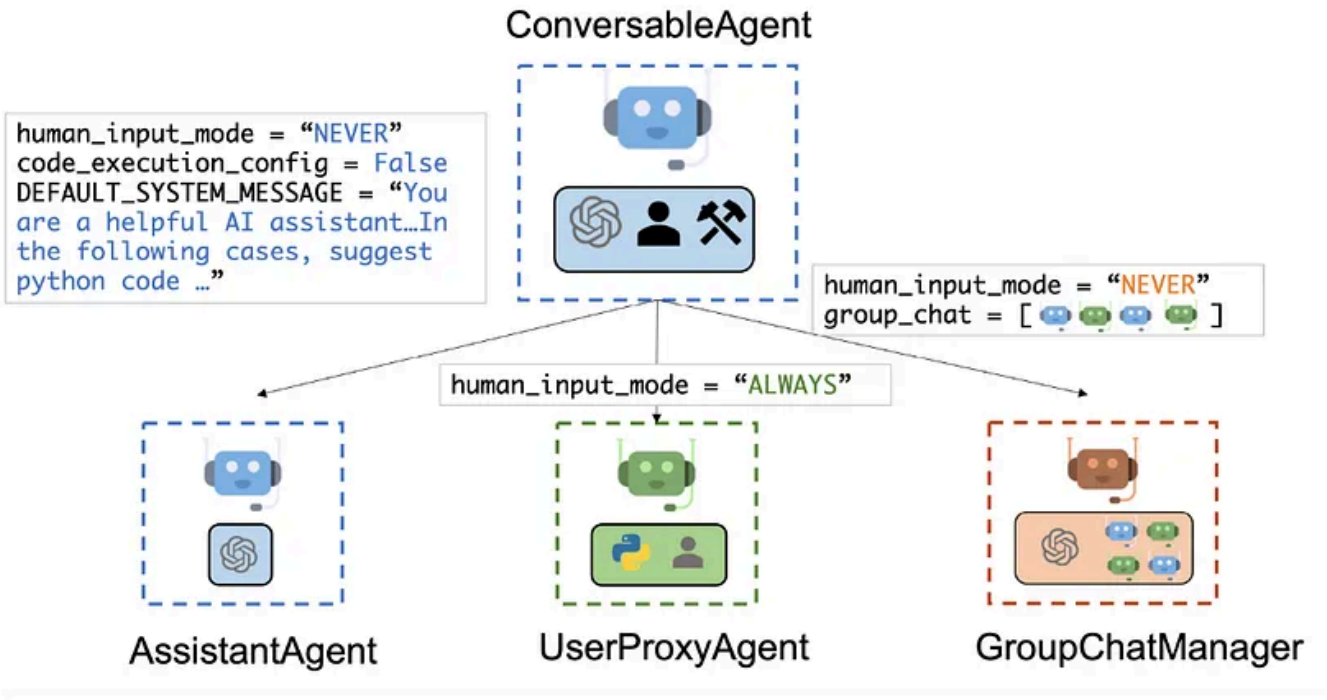
Image created by author, Background image by Hollywood reporter

Imagine a team of virtual assistants collaborating to handle customer support queries seamlessly. Each assistant specializes in a specific task, ensuring accurate, efficient, and optimized responses. This is the essence of the agentic approach in LLMs.

RAG or **Retrieval-Augmented Generation** pipelines are now integral parts of LLM applications. There are tools like Arize Phoenix, ragas, TrueLens, etc. that use a wide variety of metrics for the evaluation of RAGs. After the advancements in RAG pipelines, the **Agentic approach** has become a new approach for developing LLM applications. Everyone is eager to convert their existing or new products into agentic workflows. It's exciting to see fully capable LLMs who can interact with each other, engage in proper group chats, and collaboratively arrive at optimized and comprehensive solutions, with or without human intervention.

What are agents?

Agents are orchestration platforms or tools in LLMs, designed to combine multiple LLMs or even with no LLMs to perform tasks with little to no human intervention. Each agent works autonomously on individual tasks but also can discuss, ask, brainstorm, and refine their work. We can use any LLM to create an agent and to do any task.



source: microsoft Autogen, This image represents types of agents and its aspect in autogen agent library

What is the agentic approach?

The agentic approach involves breaking down a workflow into smaller, autonomous agents that collaboratively solve problems. Instead of having a monolithic system handle all tasks, this approach distributes responsibilities among specialized agents, enabling more efficient and scalable solutions. It is an approach to convert the task or orchestration into individual agents like a workflow. In simple words, assigning different tasks to different people and telling them to collaborate and create a refined solution via group discussion, just like we do in our brain-storming sessions.

> *Example:*

Let's consider we are going to build a simple LLM-based question-answering system.

In **LLM RAG pipelines**, we have a retriever who retrieves the most similar data on the basis of the provided question. Then those top chunks and question we send to LLM and LLM generates an answer from that. if we want any refinement or change in answer in terms of answer structure or content, then we simply tweak the prompt, then again we will do the same procedure and check the answer.

In terms of the **agentic approach**, on the basis of the above case, what I can do is, I can make 3 agents one will be assigned with tasks of retriever, that will be **retriever agent**, one will be responsible for generating answer, that will be an **answer generator agent** and we can put one agent that will be to giving feedback on the generated answer, that we can put as **answer validator agent**.

Now after the retriever gives relevant chunks, the answer generator agent and answer validator agent will have a conversation and after multiple rounds of feedback and back and forth, the proper answer will get generated. I can set proper prompts for these two agents to specify their tasks. Also, I won't give access to LLM to the retriever agent because it doesn't need that, it just wants to do retrieval autonomously.

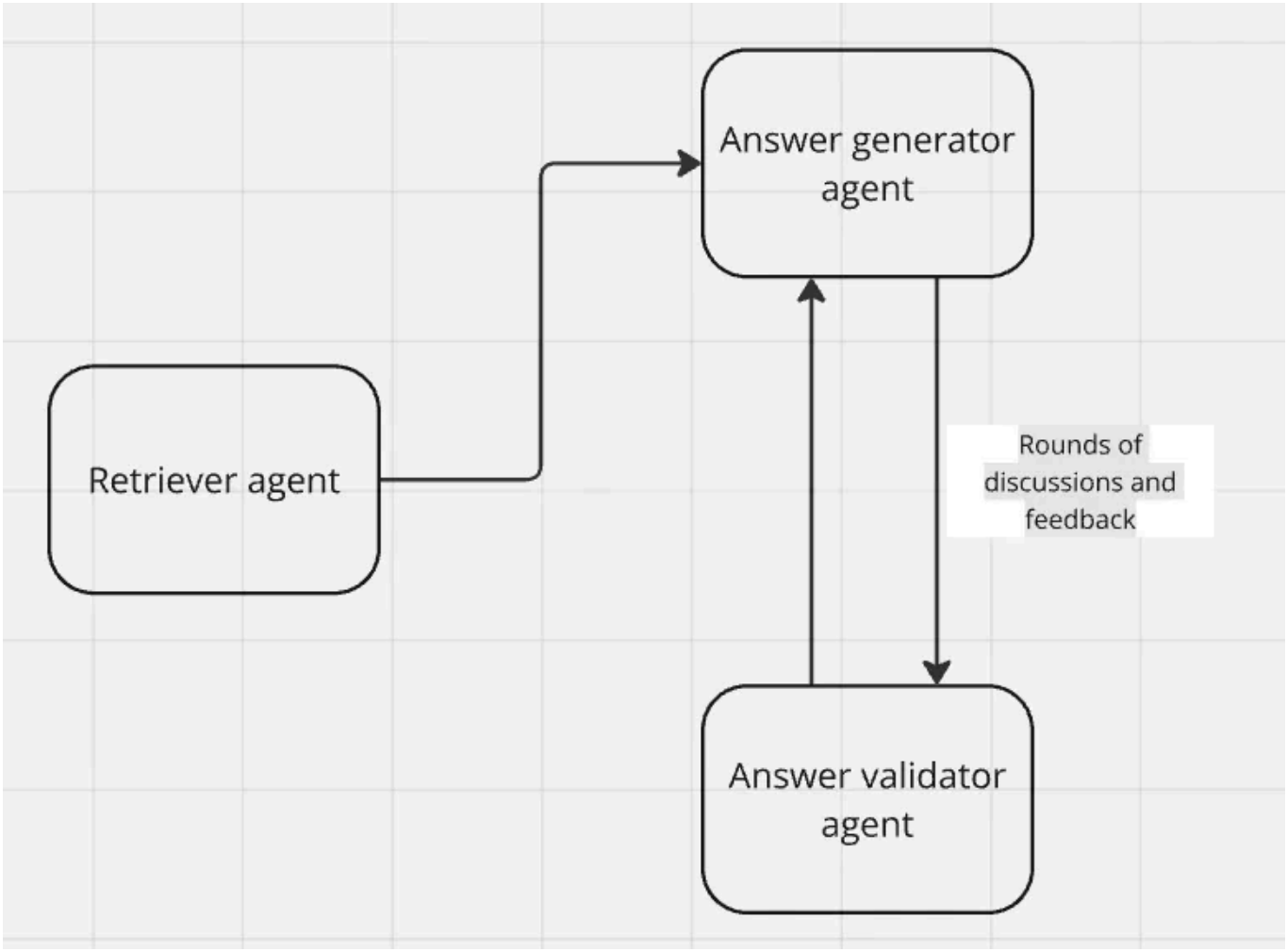


Image by author

The above image shows showing exact procedure that I mentioned in the above paragraphs. This is how we can use agents for autonomous tasks.

Evaluation

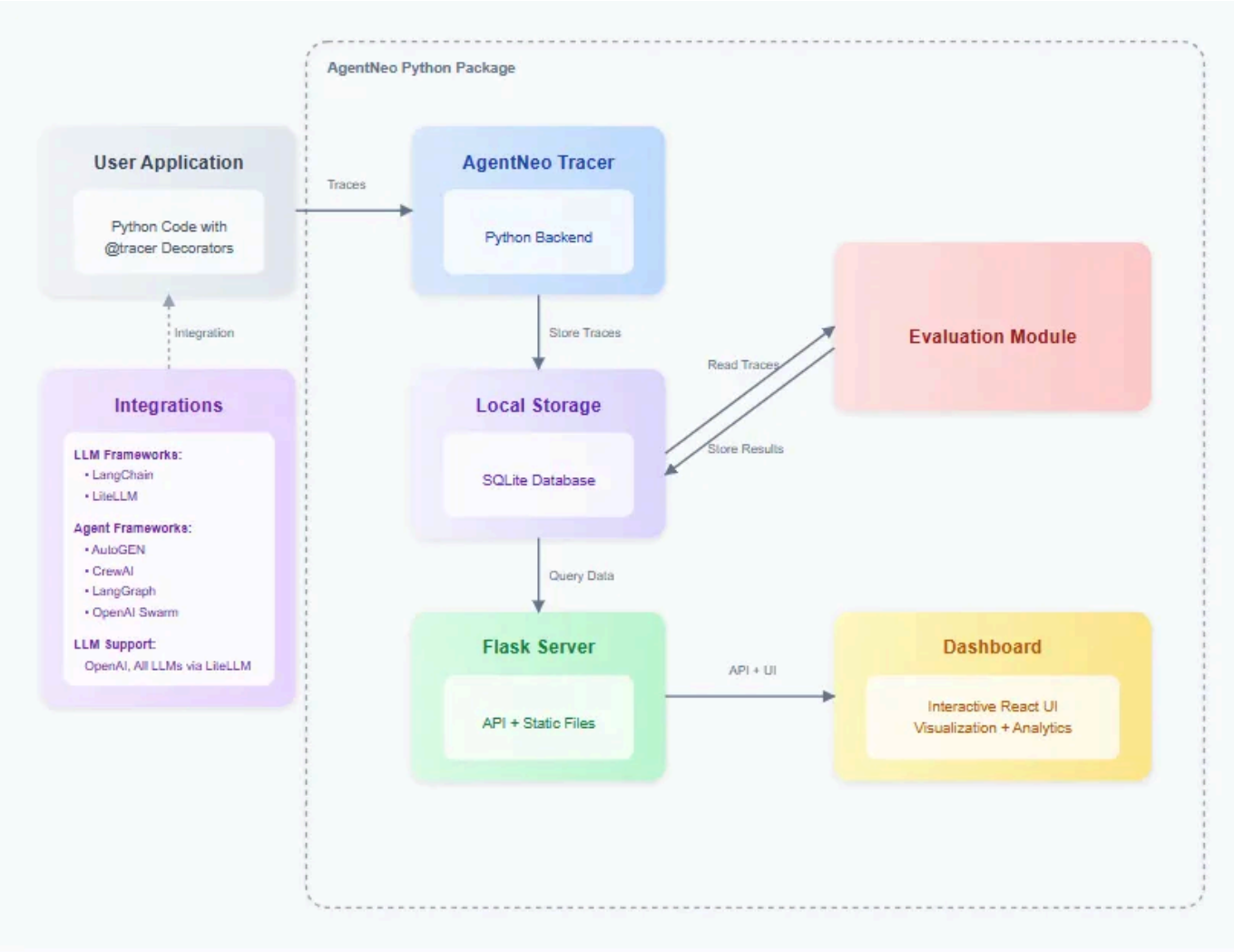
After understanding the agents, we can now move towards evaluation of agents.

In ML or DL, we have **discrete outputs** (mostly speaking) so their evaluation metrics are somewhat fixed or easily generalizable. But when it comes to the RAG pipeline or agent's evaluation, then the interesting thing starts. RAG evaluation utilizes a rich diversity of metrics like context relevancy, faithfulness, context recall, recall, toxicity, Recall@K, Precision@K, etc.

Agents rely on specific metrics to evaluate their performance effectively. Many of the metrics used for traditional systems, such as context relevancy, faithfulness, and recall, can also be applied to agents. Additionally, there are specialized tools designed to streamline the evaluation process for agents.

> *Let's start with,*

1. Agentneo



Source: Agentneo official repository

Agentneo is launched by the company **ragaAI**. It is a Python SDK for Agent AI Observability, Monitoring, and Evaluation Framework. It offers features

such as tracing agents, LLMs, and tools, debugging multi-agent systems, a self-hosted dashboard, and advanced analytics with timeline and execution graph visualization.

> *Start with,*

```
!pip install agentneo
```

With this simple installation, you can have an agentneo installed in your environment.

The next task is so simple. start the **tracing** [in simple terms, logging] after the initialization of agents and before the group chat or conversation starts.

```
neo_session = AgentNeo(session_name="my_session")
neo_session.create_project(project_name="my_project")

tracer = Tracer(session=neo_session)
tracer.start()
```

This will start a tracing of agentic conversation. Agentneo provides a custom dashboard like many evolution apps to see all aspects of agents.

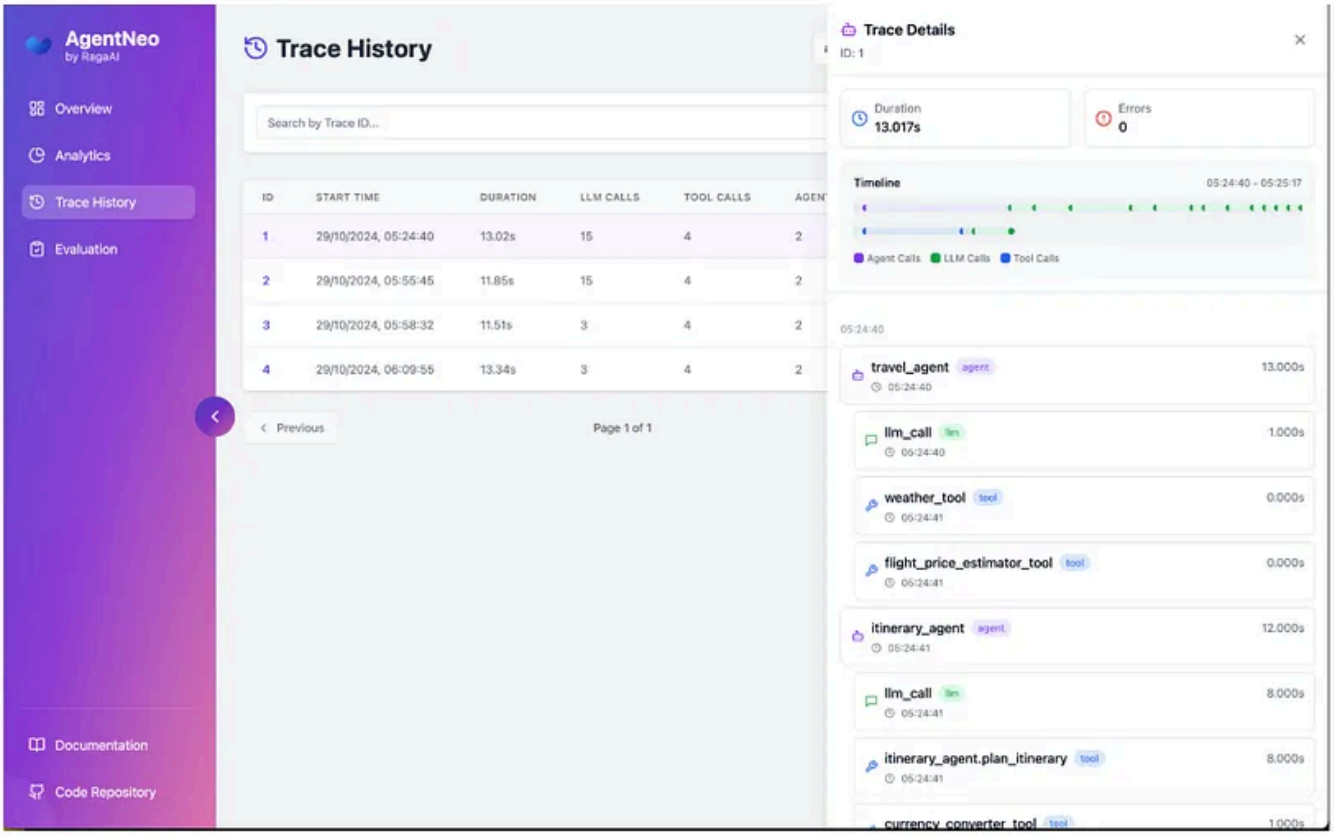
> *You can evaluate performance and set metrics according to your requirements.*

```
exe = Evaluation(session=neo_session, trace_id=tracer.trace_id)

# run a single metric
exe.evaluate(metric_list=['metric_name'])

# get your evaluated metrics results
metric_results = exe.get_results()
print(metric_results)
```

> *After the group chat and discussion, terminate the tracing and launch the dashboard.*

```
tracer.stop()

launch_dashboard(port=3000)
```

> *you can access the interactive dashboard by visiting `http://localhost:3000` in your web browser.*



Dashboard for Agentneo

In this dashboard, you can trace many things as analytics, time, LLM calls, and the main thing that is very important in **agetic conversation**, the chat of agents.

It has integration with many agentic libraries such as Autogen and Crewai and many are planned in the future. I have used it with Autogen and it is a nice tool to use for tracing agents.

Github: Agentneo | RagaAI

2. Arize phoenix

Source: Official Arize phoenix website

Arize Phoenix is an **Open-source LLM tracing** and **evaluation too**l. It helps evaluation using the below aspects.

> **Tracing** helps get detailed insights into the execution flow, thereby making it easier to identify & resolve issues. Provides metrics, logs and metadata.
>
> **Prompt Tracking**: Provides a separate space to create, manage, and experiment with prompt variations. More info here .
>
> **Embeddings Visualizer**
>
> **Evals Testing & Benchmarking:** Define custom metrics, collect user feedback, and leverage separate LLMs for automated assessment. Phoenix offers tools for analysing evaluation results, identifying trends, and tracking improvements. Benchmark these evaluation metrics against industry standards or custom baselines.
>
> **Curate Data**: Provides tools for data exploration, cleaning & labelling

Phoenix is also used to trace the agentic approaches. Phoenix instruments Autogen by instrumenting the underlying model library it's using.

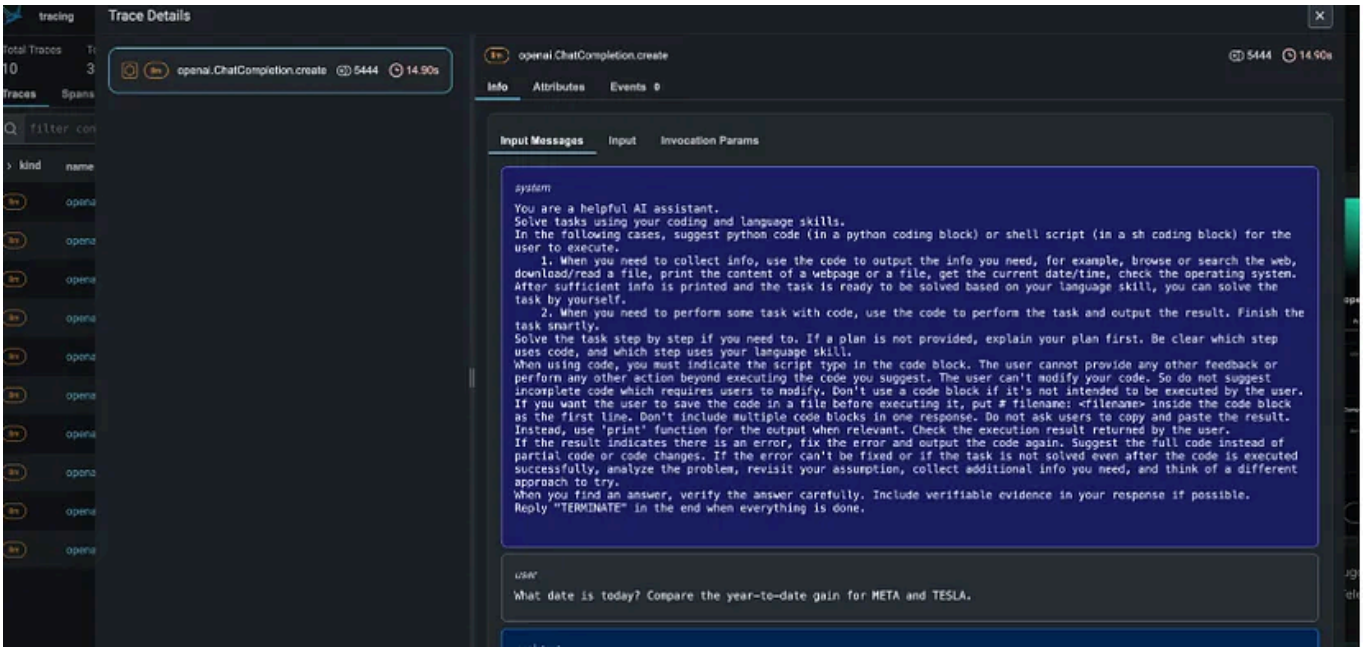
> *You can start by installing libraries.*

```
!pip install -qq arize-phoenix arize-phoenix-otel ipython matplotlib pycm scikit
```

> *Below is the code to execute it locally*

```
from openinference.instrumentation.openai import OpenAIInstrumentor

import phoenix as px
from phoenix.otel import register

px.launch_app()
tracer_provider = register()
OpenAIInstrumentor().instrument(tracer_provider=tracer_provider)
```

> *Below is the dashboard that you can see after the execution*



I have used Agentneo but honestly didn't use Phoenix as much the reason for this? the time when I was trying to use Phoenix, it wasn't supporting any Google's models and it had limited access to Openai's models. I know they might have now included all the models. Below are the links to the notebook for instructions related to the procedure of using Phoenix with Autogen.

Notebook | **Procedural Phoenix** | **LLM evaluation**

3.TruLens

In many applications, to trace and monitor the RAG pipeline, I have used **TrueLens.**

Source: TrueLens github repository

TruLens is a software tool that helps you to objectively measure the quality and effectiveness of your LLM-based applications using feedback functions. Feedback functions help to programmatically evaluate the quality of inputs, outputs, and intermediate results so that you can expedite and scale up experiment evaluation. Use it for a wide variety of use cases including question answering, summarization, retrieval-augmented generation, and agent-based applications.

TruLens can evaluate your LLM app with the following kinds of feedback functions to increase performance and minimize risk:

> *Context Relevance*
>
> *Groundedness*
>
> *Answer Relevance*

> *Comprehensiveness*
>
> *Harmful or toxic language*
>
> *User sentiment*
>
> *Language mismatch*
>
> *Fairness and bias*
>
> *Or other custom feedback functions you provide*

Image credit : Official TrueLens website

On their official website, they mentioned the usage of TrueLens in the agent's evaluation. If you want to use TrueLens then it is as easy as other tools.

> *Install dependencies*

```
from trulens.apps.custom import instrument
from trulens.core import TruSession

session = TruSession()
session.reset_database()
```

> *Set what metrics you want*

```
import numpy as np
from trulens.core import Feedback
from trulens.core import Select
from trulens.providers.openai import OpenAI

provider = OpenAI(model_engine="gpt-4")

# Define a groundedness feedback function
f_groundedness = (
    Feedback(
        provider.groundedness_measure_with_cot_reasons, name="Groundedness"
    )
    .on(Select.RecordCalls.retrieve.rets.collect())
    .on_output()
)
# Question/answer relevance between overall question and answer.
f_answer_relevance = (
    Feedback(provider.relevance_with_cot_reasons, name="Answer Relevance")
    .on_input()
    .on_output()
)

# Context relevance between question and each context chunk.
f_context_relevance = (
    Feedback(
        provider.context_relevance_with_cot_reasons, name="Context Relevance"
    )
    .on_input()
    .on(Select.RecordCalls.retrieve.rets[:])
    .aggregate(np.mean)  # choose a different aggregation method if you wish
)
```

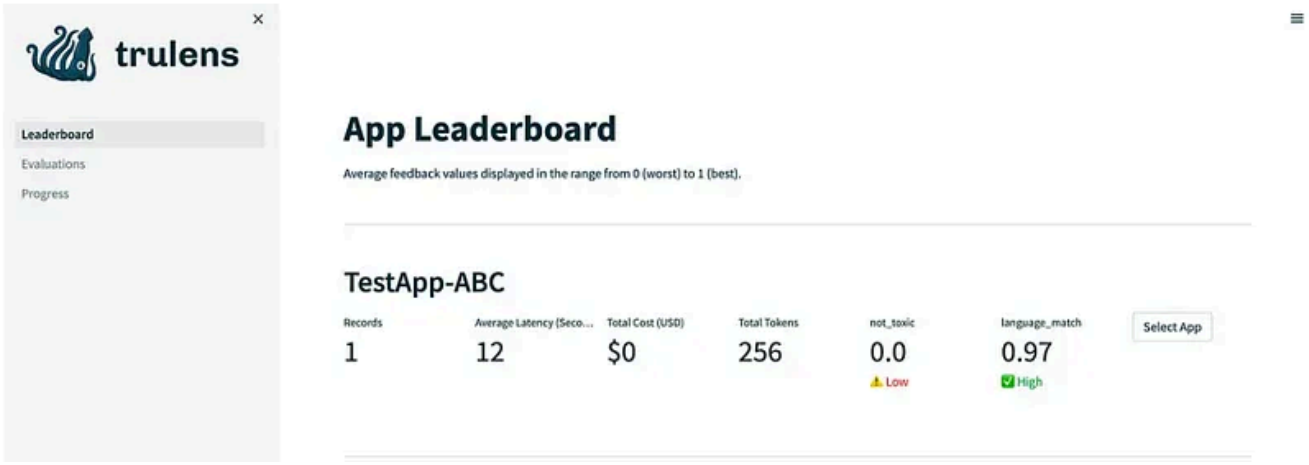
> *Construct the app*

```
from trulens.apps.custom import TruCustomApp

tru_rag = TruCustomApp(
    rag,
    app_name="RAG",
    app_version="base",
    feedbacks=[f_groundedness, f_answer_relevance, f_context_relevance],
)
```

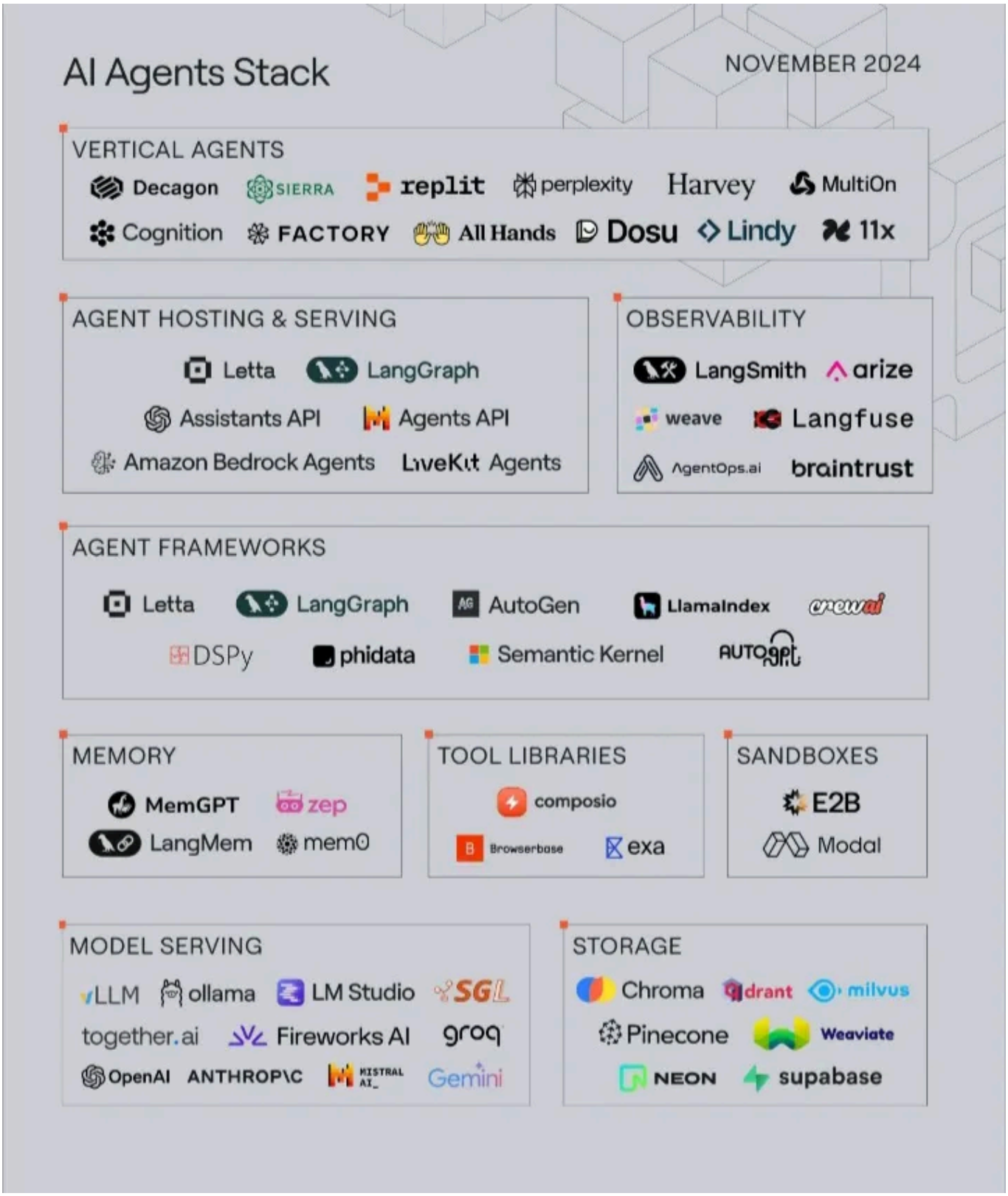
> *We can view results in the leaderboard. Check and run the dashboard*

```
session.get_leaderboard()
from trulens.dashboard import run_dashboard

run_dashboard(session)
```

> *After that, you can see the dashboard similar to the below image.*



I didn't include all code but **THIS COLAB NOTEBOOK** will guide you to set up and execute the TrueLens app.



Pic credits: Adam Silverman

Final words

All mentioned tools have their own UI and you can see it in your localhost. In my opinion, if there are many agents in your application, then tracing is very important. It is dependent on the complexity and needs of your application which things you want to trace and monitor. I mostly find the conversation part more helpful while evaluating because it gives us an idea of which agent is doing what and on that basis, which prompt we need to tune and it is also helpful to decide the number of conversation rounds. If you know any other tools to evaluate agents then comment below so that I can try and add it to this blog.

If you have found this article insightful

It is a proven fact that "**Generosity makes you a happier person**"; therefore, Give claps to the article if you liked it. If you found this article insightful, follow me on **Linkedin** and **Medium**. You can also **subscribe** to get notified when I publish articles. Let's create a community! Thanks for your support!

You can read my other blogs related to :

Weights & Activation functions | Deep Learning: 1

This blog explores the basics of weights, activation functions, their types, and commonly used activation functions

medium.com

How to Choose the Best Algorithm for Your Machine Learning Project

Optimizing ML Model Performance: A Guide to Algorithm Selection

medium.com

How you can improve output masks for segmentation projects?

Problems with segmentation masks and step-wise solutions to improve output masks with GitHub discussions

medium.com

Comprehensive Guide: Top Computer Vision Resources All in One Blog

Save this blog for comprehensive resources for computer vision

medium.com

Signing off,

CHINMAY!

Large Language Models | Agents | Artificial Intelligence | AI | OpenAI

Published in Towards AI

62K Followers · Last published 2 hours ago

Follow

The leading AI community and content platform focused on making AI accessible to all

Written by Chinmay Bhalerao

1.6K Followers · 123 Following

Follow

AI-ML Researcher & Developer | 3 X Top writer in Artificial intelligence, Computer vision & Object detection | Mathematical Modelling & Simulations

More from Chinmay Bhalerao and Towards AI

In Data And Beyond by Chinmay Bhalerao

How Filter Bubbles Are Biasing Your Opinions on Social Media

Filter Bubbles and Echo Chambers: How Social Media is Biasing Our Opinions and the...

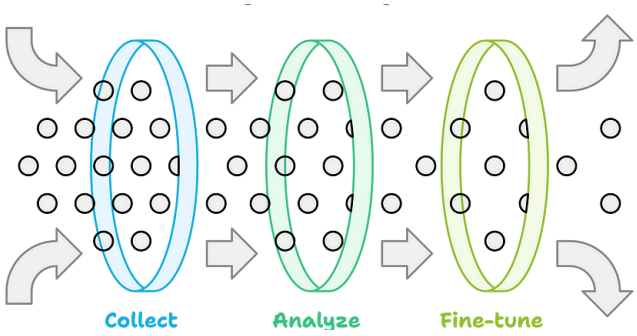
Jul 8, 2023 303 2

In Towards AI by Mohit Sewak, Ph.D.

LLM Agent Jailbreaking and Defense—101

The Complete Guide to LLM Agent Security: Ways to Secure Your GenAI Agents

1d ago 213 1

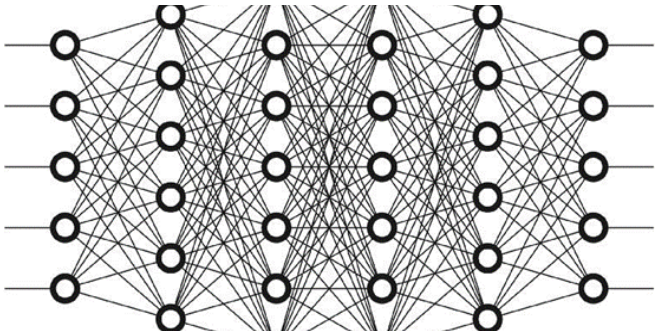


In Towards AI by Surya Maddula

Feedback Loop Mechanisms in Retrieval Systems

Overview: Examine systems that learn from user interactions to enhance future retrieval...

5d ago 256 2



In Data And Beyond by Chinmay Bhalerao

From Overfitting, underfitting to Dropout and Beyond| Deep...

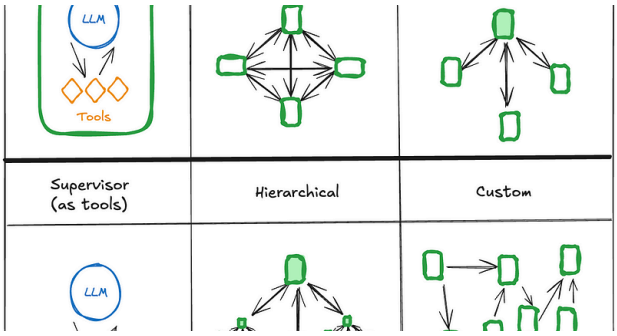
This blog explores the basics of CNN, RNN, and basic concepts like overfitting,...

Oct 26 394 1

See all from Chinmay Bhalerao

See all from Towards AI

Recommended from Medium



In Level Up Coding by Md Monsur ali

OmniVision-968M: The World's Most Compact and Smallest...

How OmniVision-968M Outperforms Existing Vision-Language Models and Enables...

Nov 17 370 4

Alfredo Sone

AI Agents: How to build Digital Workers

Key learnings to understand and design intelligent digital workers.

Nov 18 368 9

Lists

AI Regulation

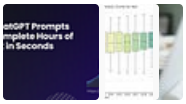
6 stories · 633 saves

Generative AI Recommended Reading

52 stories · 1526 saves

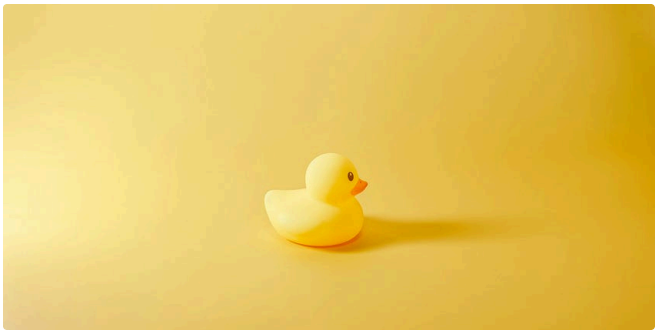
Natural Language Processing

1839 stories · 1461 saves



ChatGPT prompts

50 stories · 2297 saves

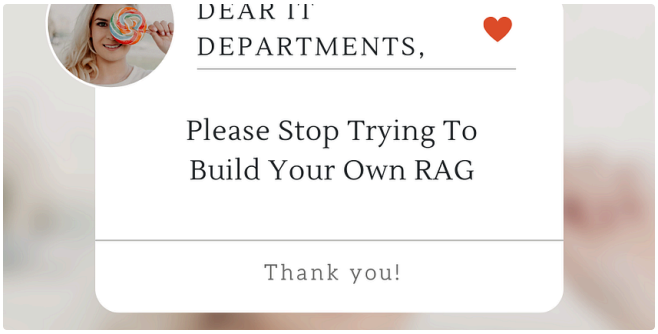


In Towards Data Science by Dr. Leon Eversberg

Improved RAG Document Processing With Markdown

How to read and convert PDFs to Markdown for better RAG results with LLMs

Nov 19 597 4



In Towards AI by Alden Do Rosario

Dear IT Departments, Please Stop Trying To Build Your Own RAG

IT departments convince themselves that building their own RAG-based chat is easy. It'...

Nov 11 3.4K 79

In Cubed (we have moved to differ.... by Michael ...

The Insanity of Relying on Vector Embeddings: Why RAG Fails

In RAG, the goal is to locate the stored information that has the highest percentage...

Nov 21 834 21

In Stackademic by Crafting-Code

I Stopped Using Kubernetes. Our DevOps Team Is Happier Than Ever

Why Letting Go of Kubernetes Worked for Us

Nov 19 2.9K 94

See more recommendations