A Comprehensive Guide to Agentic AI

Debmalya Biswas, PhD

Introduction to AI Agents

Reference Architecture

Agents Discovery & Marketplace

Agent Observability & Memory Management

Agentic AI Scenarios: Agentic RAGs Reinforcement Learning Agents

Responsible AI Agents

Introduction to Agentic AI

AI Agents

In the Generative AI context, Agents are representative of an **Autonomous Agent** that can execute complex tasks, e.g.,

- - make a sale,
- - plan a trip,
- - make a flight booking,
- - book a contractor to do a house job,
- - order a pizza.

Bill Gates says today's software is still 'pretty dumb,' but believes AI will 'utterly change how we live our lives'

Published Thu, Nov 16 2023·2:49 PM EST

Cheyenne DeVon

SHARE

Agentic AI in the News

Microsoft pitches AI 'agents' that can perform tasks on their own at Ignite conference

Microsoft Ignite 2024

Why agents are the next frontier of generative AI

McKinsey, July 2024

NVIDIA AI Agents: Your New Digital Coworkers

Nvidia AI Summit, Nov 2024

Salesforce's Agentforce Is Here: Trusted, Autonomous AI Agents to Scale Your Workforce

Salesforce, Oct 2024

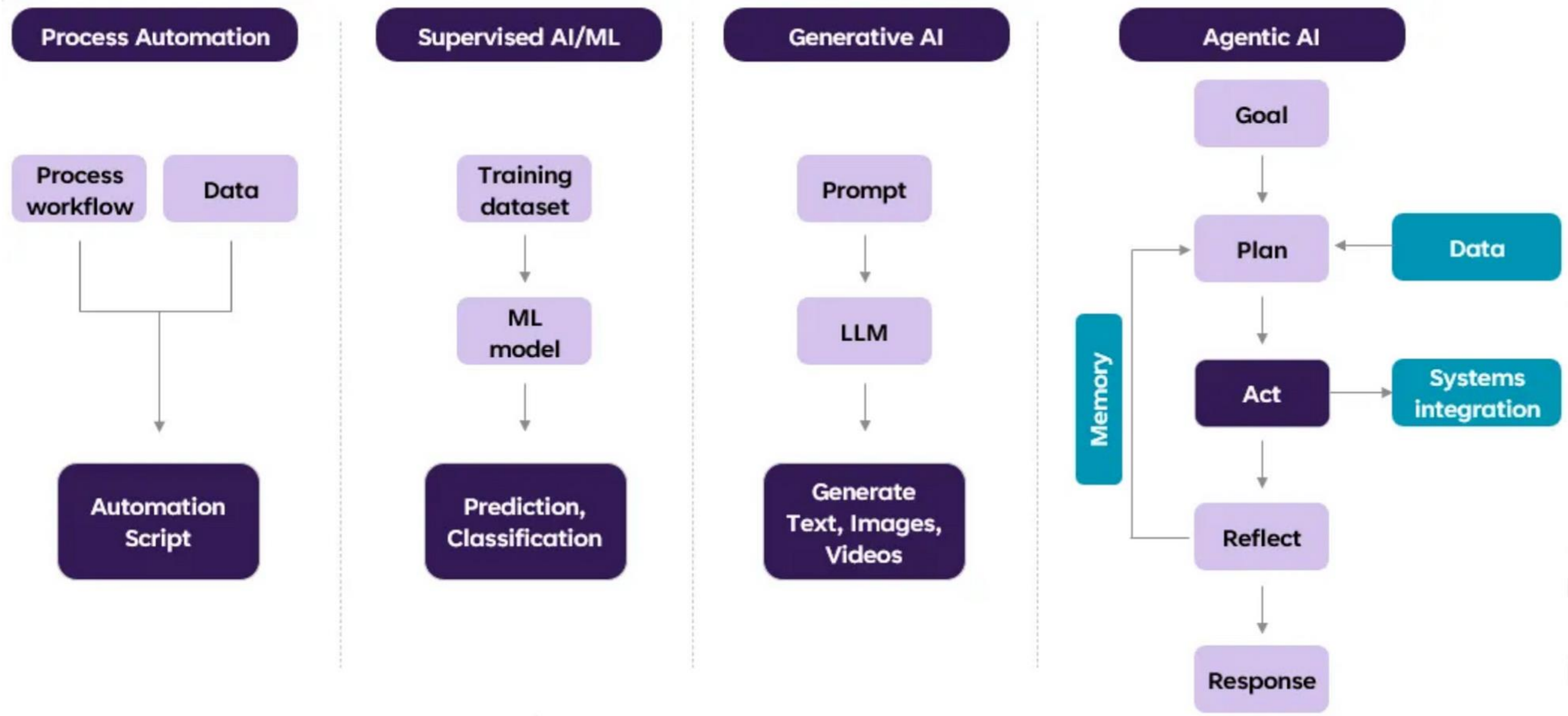
AI agents' momentum won't stop in 2025

VentureBeat, Nov 2024

ServiceNow to unlock 24/7 productivity at massive scale with AI agents for IT, Customer Service, Procurement, HR, Software Development, and more

ServiceNow, Jul 2024

Agentic AI Evolution



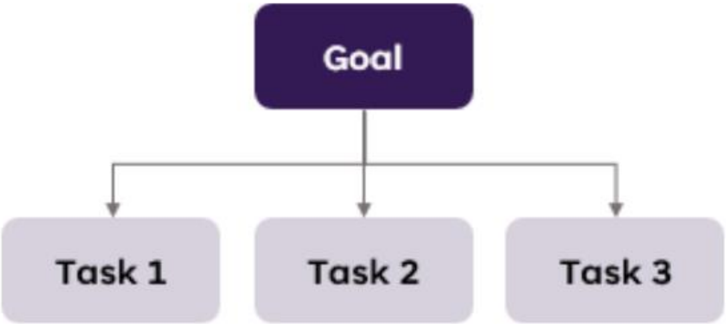
Agentic AI capabilities – Task Decomposition

Task decomposition

Given a complex user task, the system generates a plan to fulfill the request depending on the capabilities of available agents at run-time.

Chain-of-Thought (CoT)

CoT is the most widely used decomposition framework today to transform complex tasks into multiple manageable tasks and shed light into an interpretation of the model's thinking process.



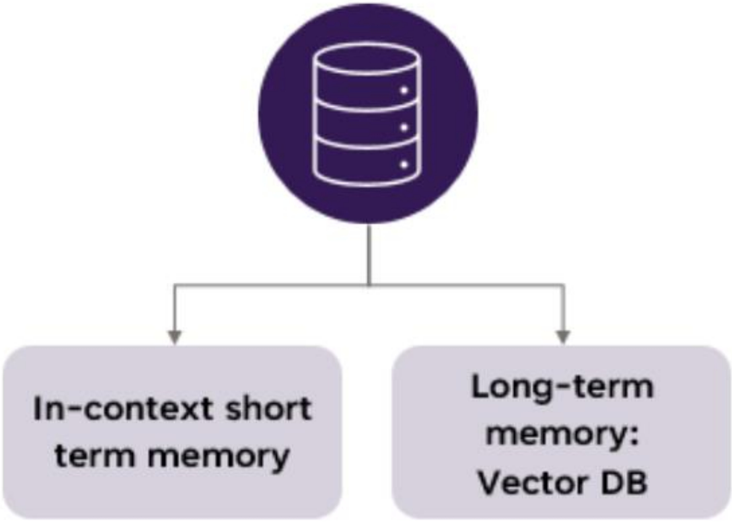
Agentic AI capabilities – Memory Management

Memory management

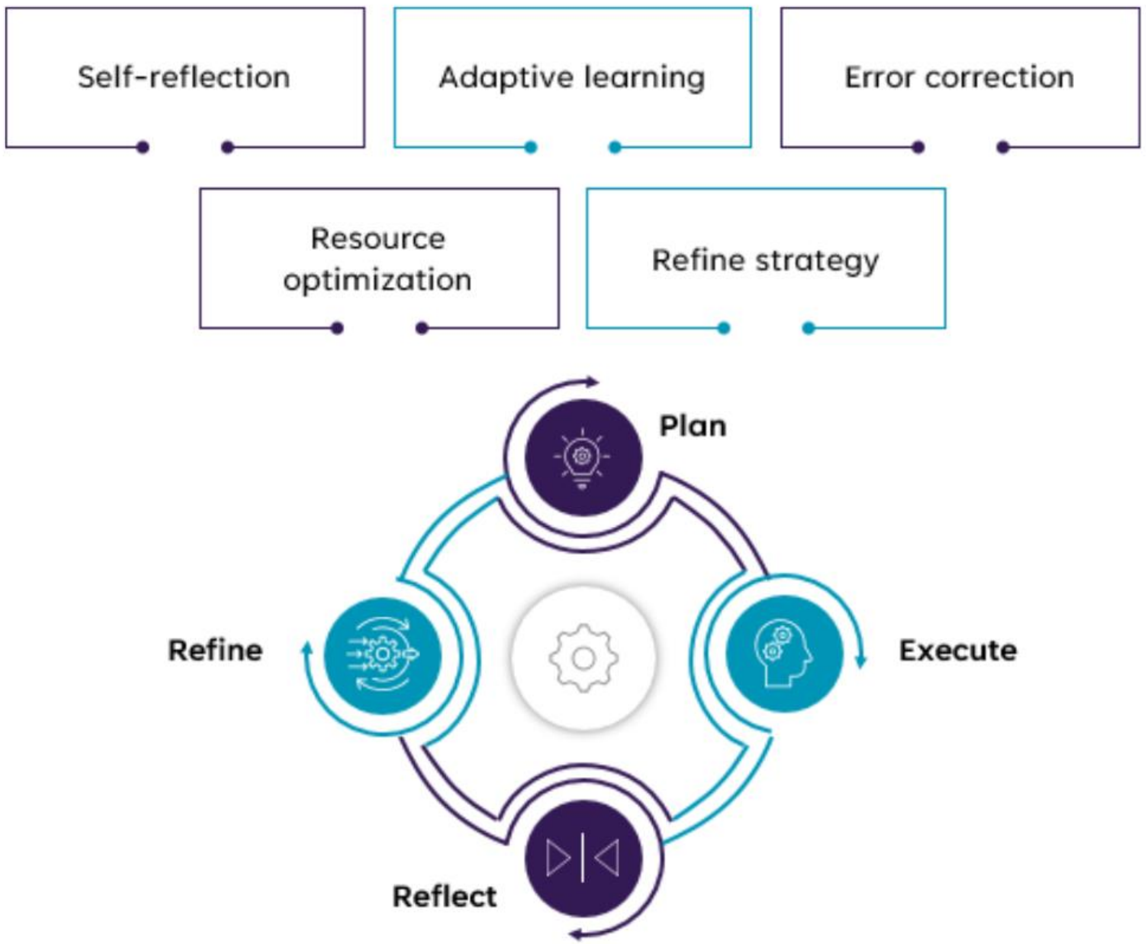
Memory management is key for Agentic AI systems for context sharing between tasks and maintaining execution context over long periods.

Long term memory

Memory management becomes challenging when agents execute for extended period as their memory is typically limited to their context. The solution is to use Vector DBs to store agent memory externally, retrieving it as needed.



Agentic AI capabilities – Reflect & Adapt



Agentic AI Use-case: Funds Email Marketing Campaign

User Query:
"Generate a tailored email campaign to achieve sales of USD 100,000 in 1 month, The applicable products and their performance metrics are available at [url]
Connect to CRM system [integration] for customer names, email addresses, and demographic details.

Agentic AI functional & non-functional capabilities

↓

Task Added: Analyze the products and performance metrics available at [url]

Task Added: Identify the target audience based on the products' performance metrics

Task Added: Create a tailored email campaign highlighting the benefits of the identified products for the target audience

Task Added: Launch and monitor the email campaign to achieve sales of USD 100,000 in 1 month

Reasoning (task decomposition)

↓

Monitor email campaign for 1 week. After 1 week, it **autonomously** decided to add the following tasks.

Long-term memory

Task Added: Find alternative products with better performance metrics to include in the email campaign

Adapt autonomously

Task Added: Utilize customer data to personalize the email with the customer's name, demographics, and highlight testimonials from other customers who have previously purchased the product.

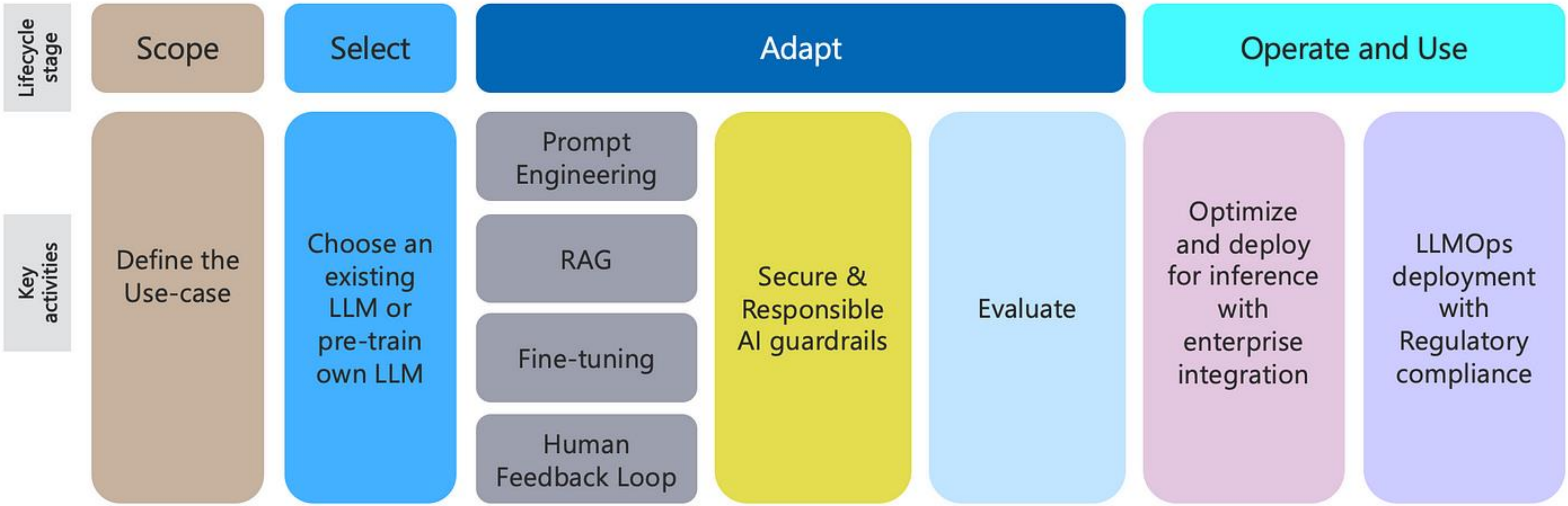
Enterprise data integration

Task Added: Perform A/B testing to further refine the email campaign

↓

Agentic AI Reference Architecture

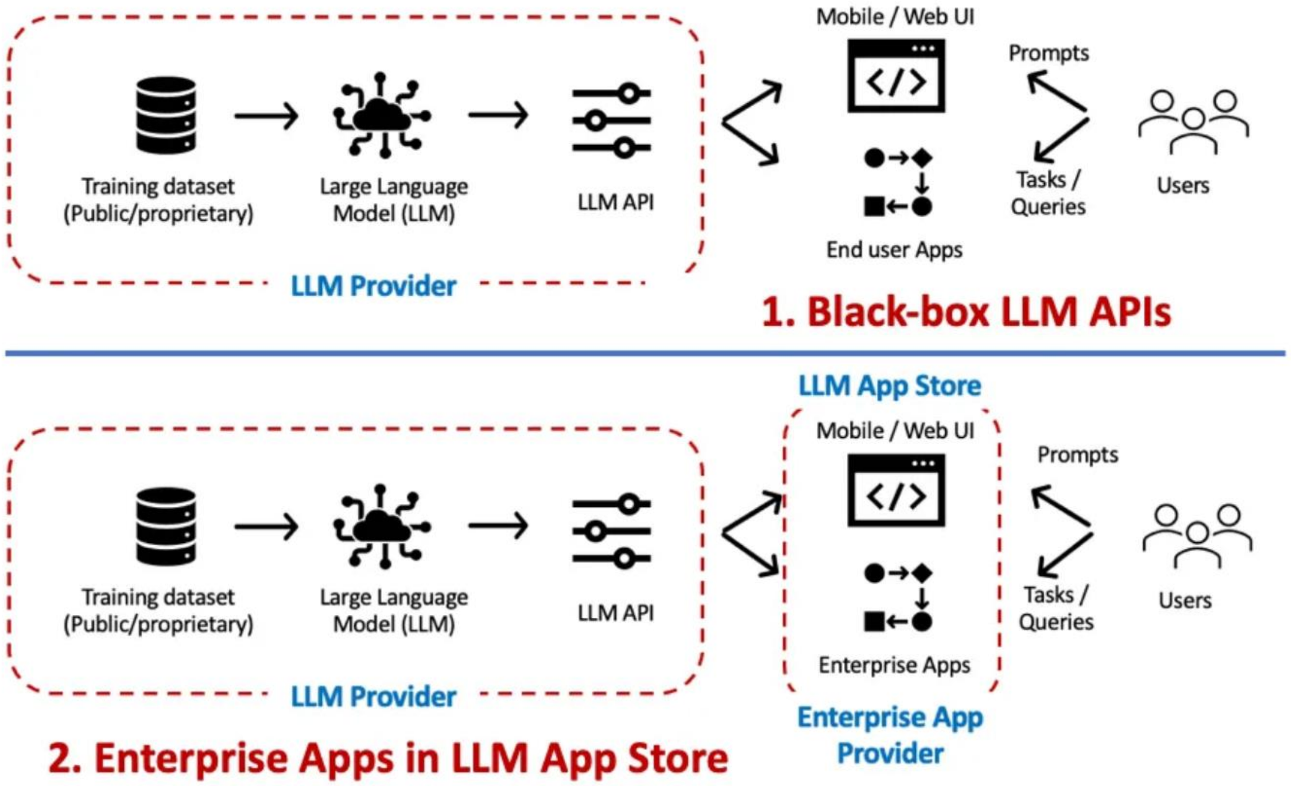
Generative AI Lifecycle



Gen AI Architecture Patterns – APIs & Embedded Gen AI

Black-box LLM APIs: This is the classic ChatGPT example, where we have black-box access to a LLM API/UI. **Prompts** are the primary interaction mechanism for such scenarios.

While **Enterprise LLM Apps** have the potential to accelerate LLM adoption by providing an enterprise ready solution; the same caution needs to be exercised as you would do before using a 3rd party ML model — **validate LLM/training data ownership, IP, liability clauses**.

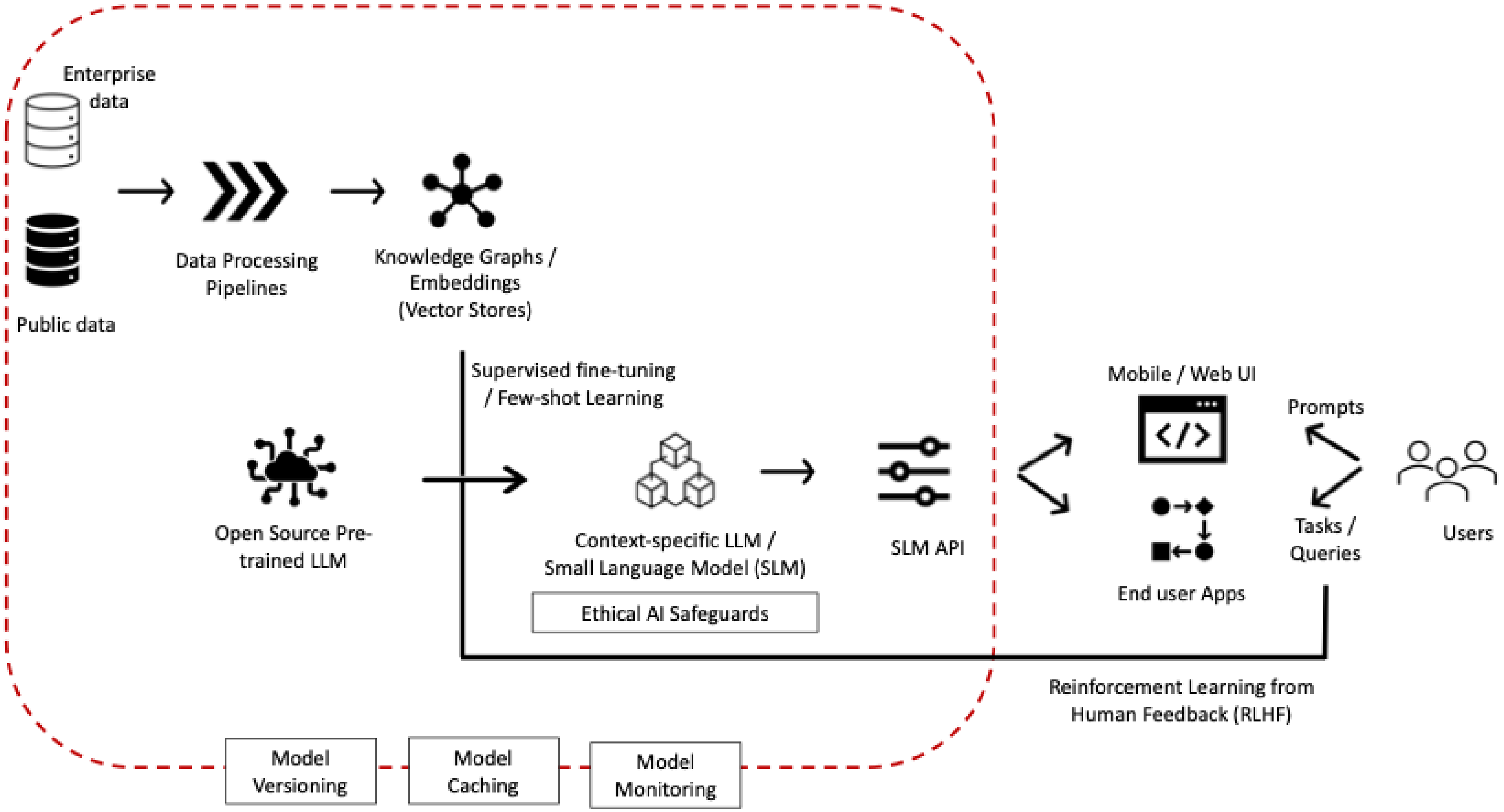


* D. Biswas. Generative AI – LLMOps Architecture Patterns. Data Driven Investor, 2023 (link)

Gen AI Architecture Patterns – Fine-tuning

LLMs are generic in nature. To realize the full potential of LLMs for Enterprises, they need to be **contextualized** with enterprise knowledge captured in terms of documents, wikis, business processes, etc.

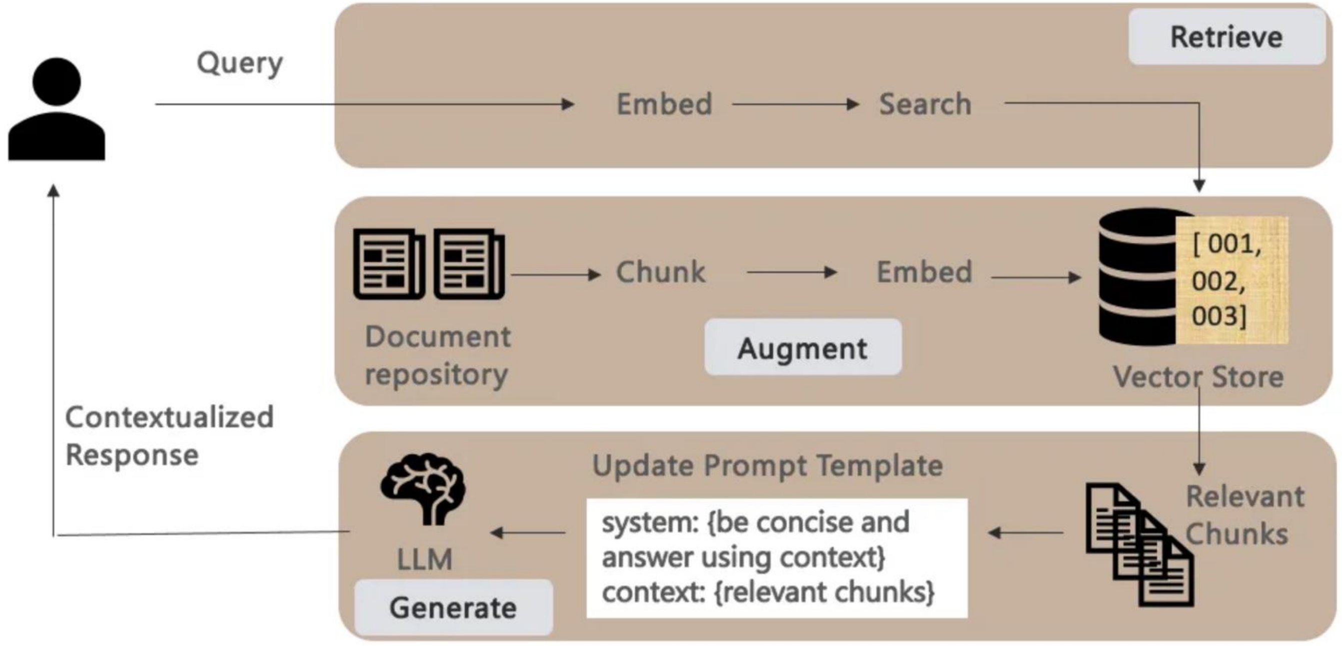
This is achieved by **fine-tuning a LLM** with enterprise knowledge / embeddings to develop a context-specific LLM.



Gen AI Architecture Patterns – Retrieval-Augmented-Generation (RAG)

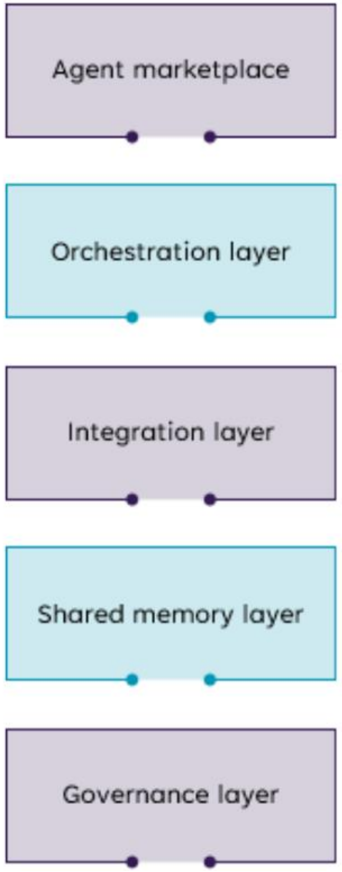
Fine-tuning is a computationally intensive process. **RAG** provides a viable alternative by providing additional context with the prompts — grounding the retrieval / responses to the given context.

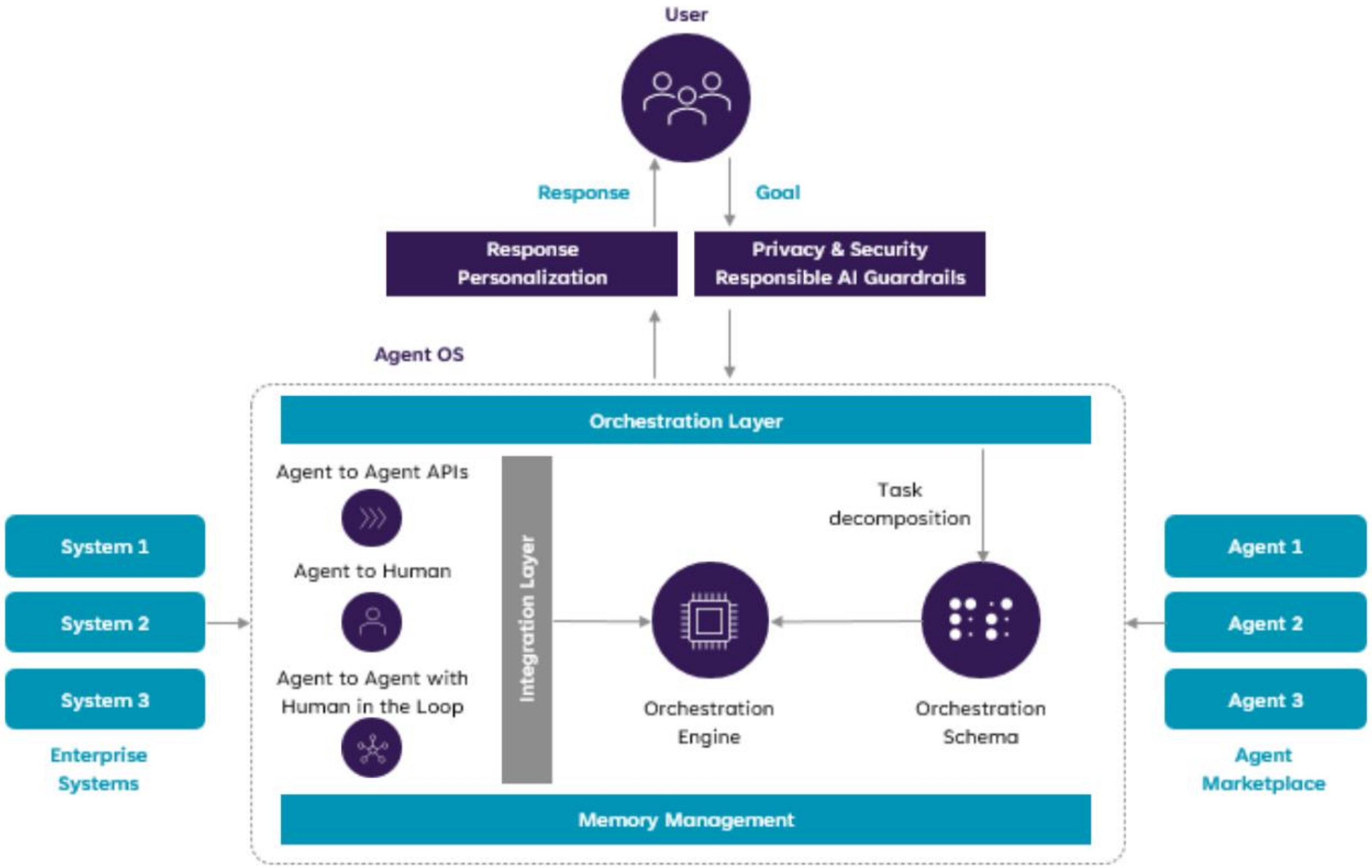
Given a user query, a RAG pipeline literally consists of the 3 phases below:
- **Retrieve**: Transform user queries to embeddings to compare its similarity score with other content.
- **Augment**: with search results / context retrieved from a vector store that is kept current and in sync with the underlying document repository.
- **Generate**: contextualized responses by making retrieved chunks part of the prompt template that provides additional context to the LLM on how to answer the query.



Agentic AI Platform Reference Architecture

The future where enterprises will be able to develop new Enterprise AI Apps by **orchestrating / composing** multiple existing AI Agents.





* D. Biswas. Stateful Monitoring and Responsible Deployment of AI Agents. 17th International Conference on Agents and Artificial Intelligence (ICAART), 2025 (link)

AI Agents Marketplace & Discovery for Multi-agent Systems

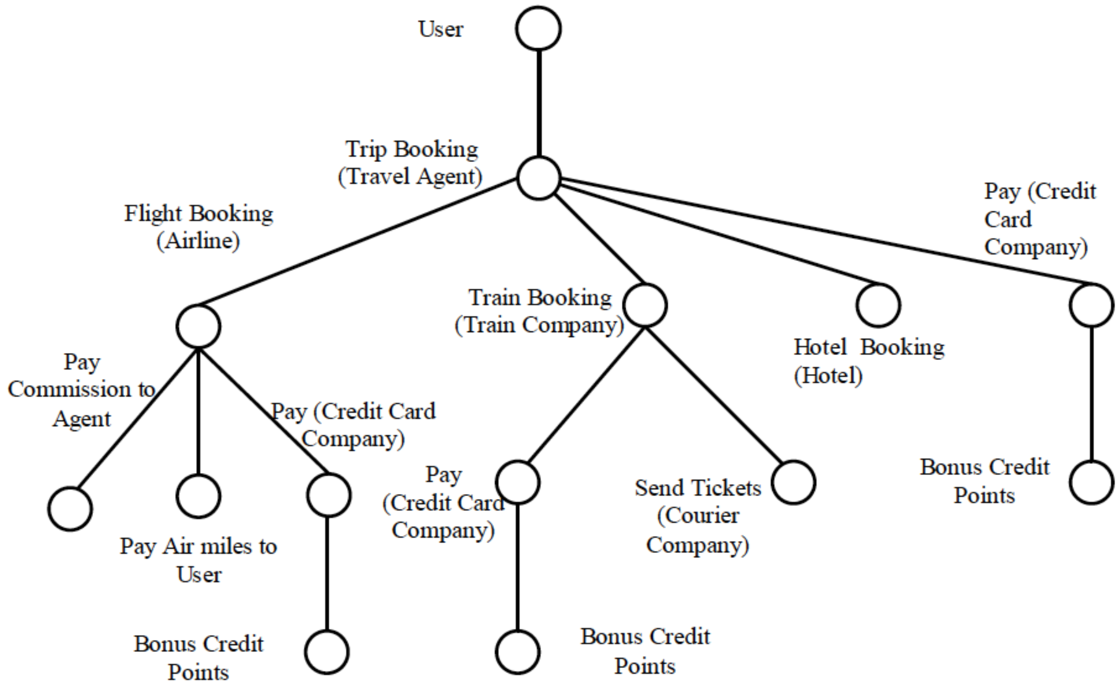
(Complex) Agentic AI Task Decomposition

A high-level approach to solving complex tasks:

- - decomposition of the given complex task into a hierarchy or workflow of) simple tasks, followed by
- - **composition** of agents able to execute the simpler tasks.

This can be achieved in a dynamic or static manner.

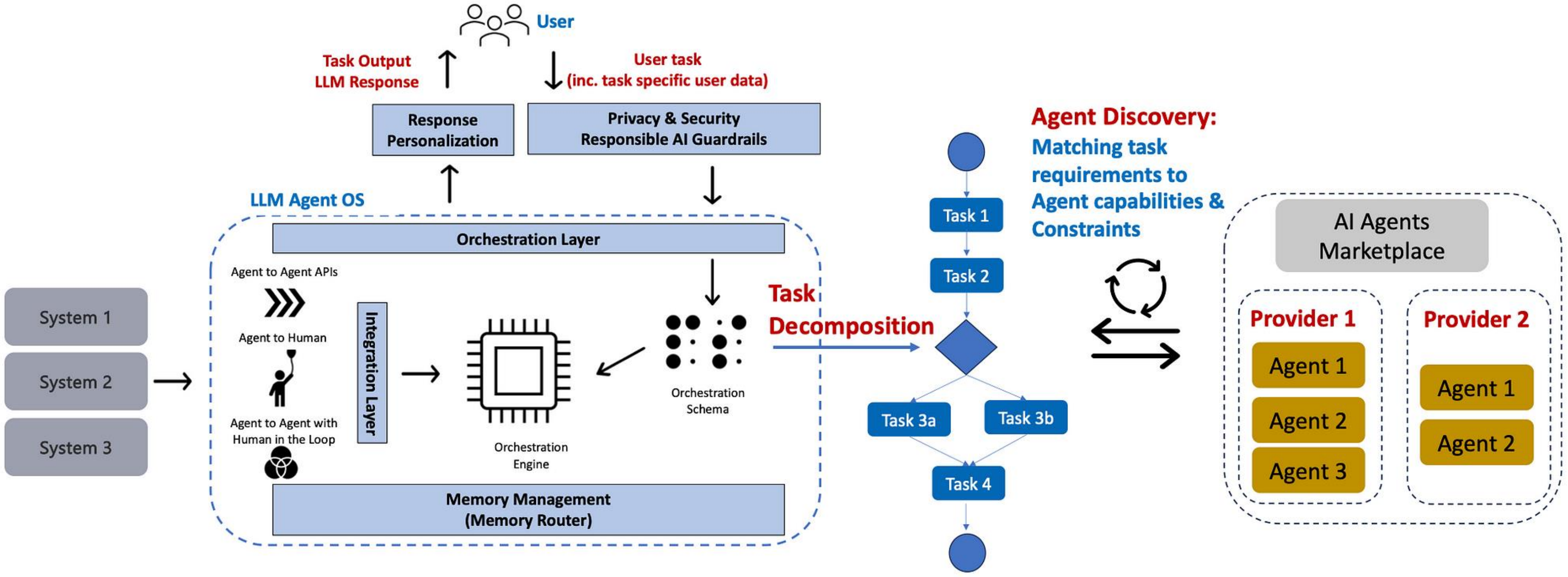
- **Dynamic**: given a complex user task, the system comes up with a plan to fulfil the request depending on the capabilities of available agents at run-time.
- **Static**: given a set of agents, composite agents are defined manually at design-time combining their capabilities.



Agent Marketplace & Discovery of AI Agents

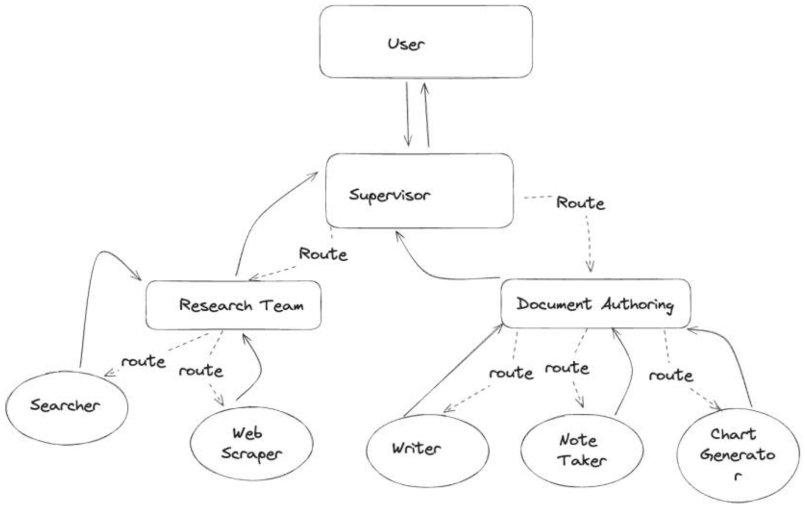
Agent decomposition and planning (be it static or dynamic) requires a **discovery** module to identify the agent(s) capable of executing a given task.

This implies that there exists a **marketplace** with a registry of agents, with a well-defined description of the agent capabilities and constraints.

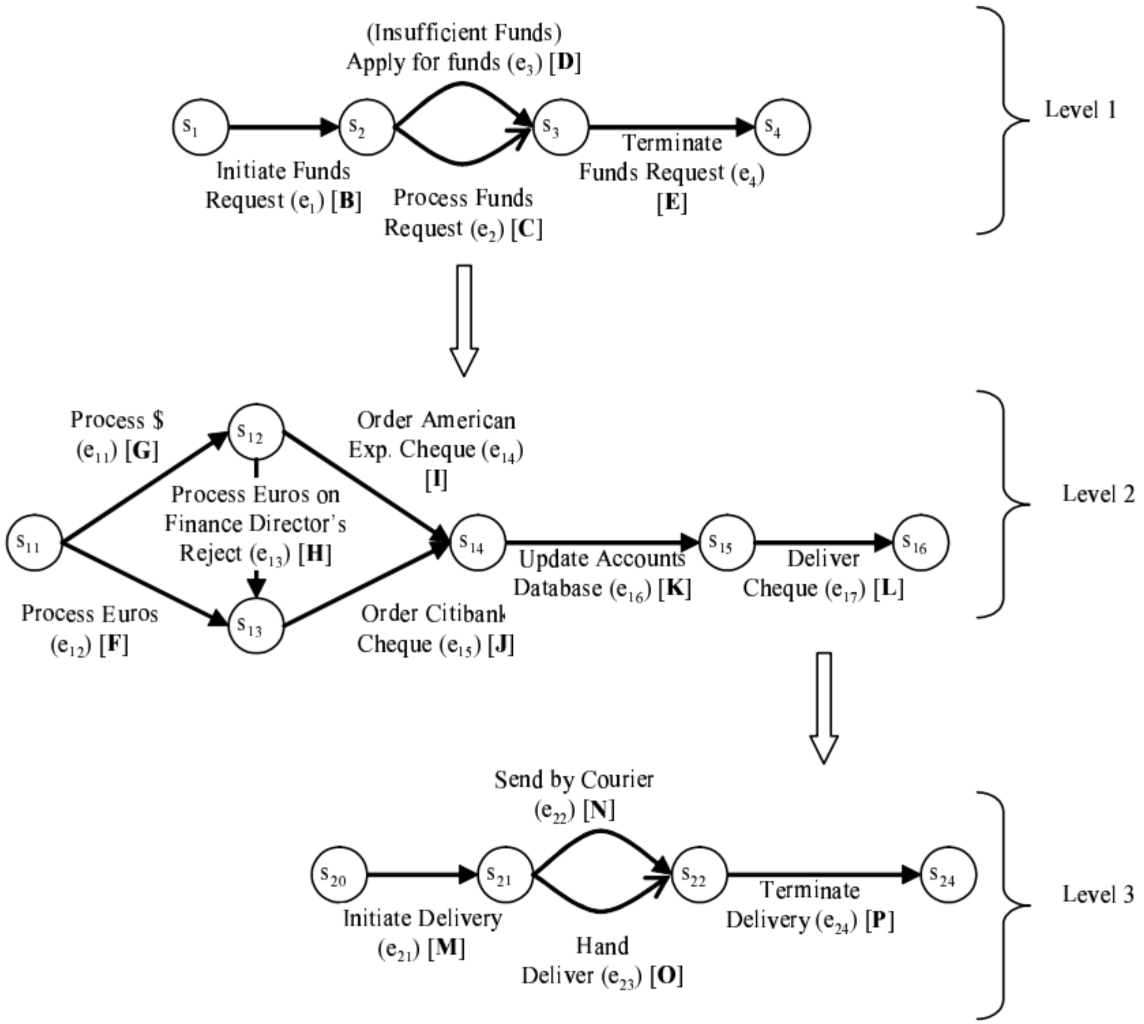


Hierarchical Agent Composition

In LangGraph (for example), **hierarchical agents** are captured as agent nodes that can be *langgraph* objects themselves, connected by **supervisor** nodes.



- LangGraph: Multi-Agent Workflows, https://blog.langchain.dev/langgraph-multi-agent-workflows/



Hierarchical Finite State Machine (FSM) representation of a Travel Funds Service

Limitations of LLMs as execution engines for Agentic AI

Current Agentic AI platforms leverage LLMs for both task **decomposition** and execution of the identified tasks / agents.

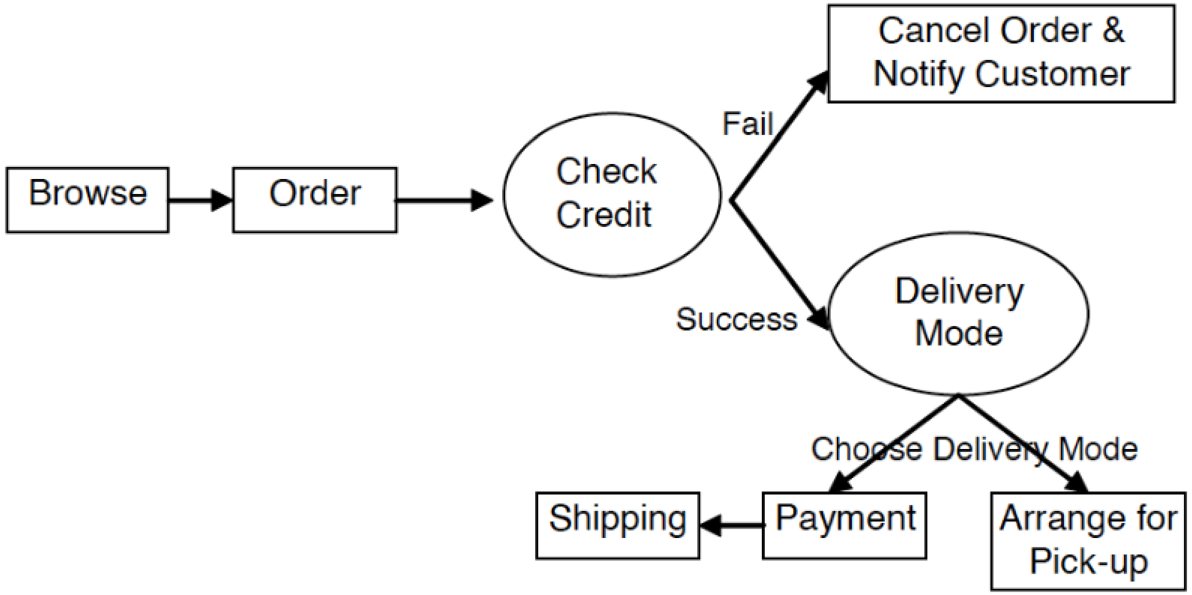
- - The overall execution occurs within the context of a single LLM, or each task can be routed to a different LLM.
- - In short, each task execution corresponds to an LLM invocation at run-time.
- Unfortunately, this approach is neither scalable nor practical for complex tasks.

LLMs cannot be expected to come-up with the most efficient (agent) **execution approac**h for a given task at run-time every time, esp. those requiring integration with enterprise systems.

Agentic AI platforms need to learn over multiple execution runs (**meta-learning**): involving a combination of user prompts, agents, and their relevant skills (capabilities).

Non-determinism in Agentic AI Systems

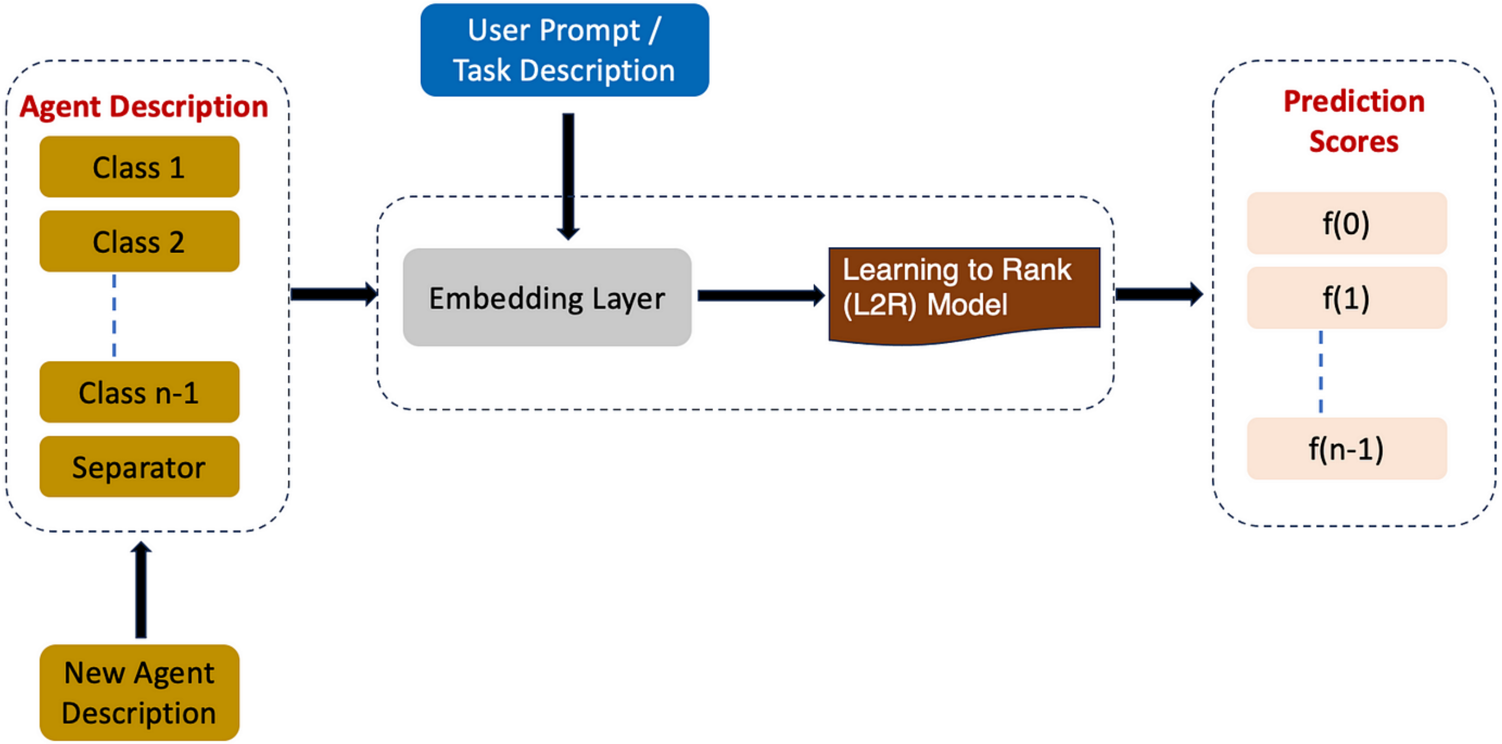
There are two non-deterministic operators in the execution plan: 'Check Credit' and 'Delivery Mode'. The choice 'Delivery Mode' indicates that the user can either pick-up the order directly from the store or have it shipped to his address. Given this, shipping is a **non-deterministic** choice and may not be invoked during the actual execution.



L2R for Agent Discovery based on Natural Language Descriptions

Learning-to-rank (L2R) algorithm to select top-k agents given a user prompt:

- We first convert agent (class) descriptions to semantic embeddings offline and use them to train the L2R model.
- The user prompts and the agents use the same generic embedding model.
- The inference results including the agent description embeddings during training and inferencing are cached to enable the meta-learning process for the L2R algorithm.



Agent Discovery based on a Constraints Model

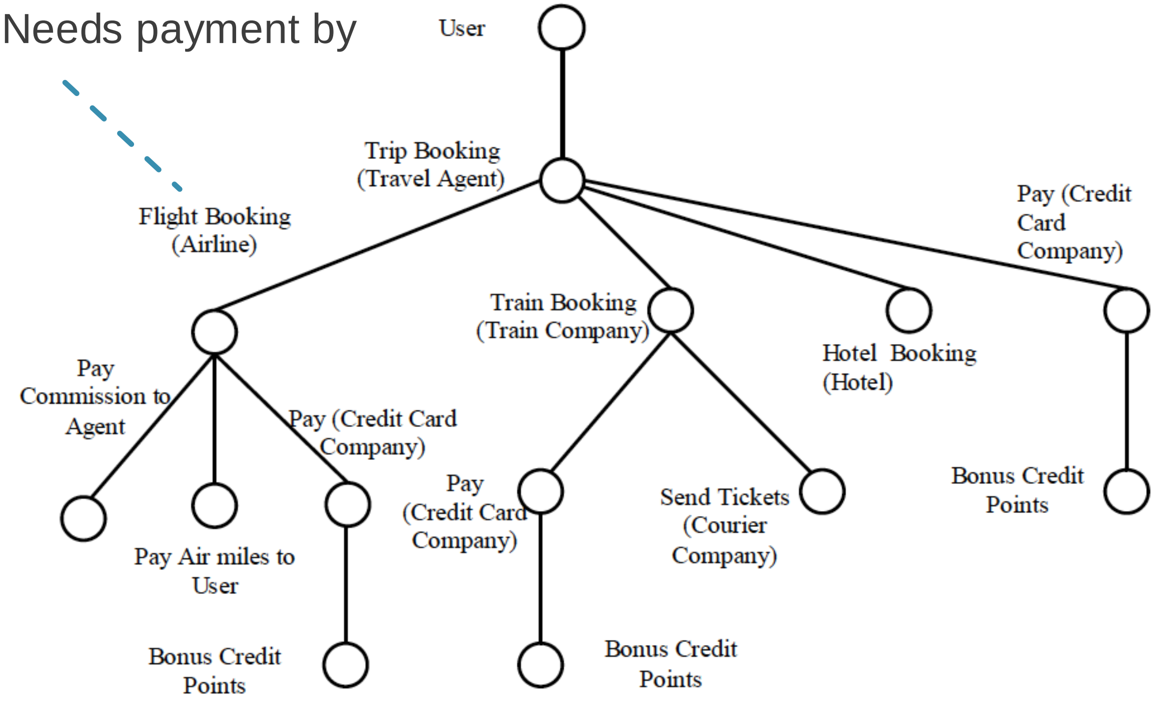
The constraints are specified as **logic predicates** in the service description of the corresponding service published by its agent.

An agent P provides a set of services $\{S_1, S_2, \dots, S_n\}$. Each service S in turn has a set of associated constraints $\{C_1, C_2, \dots, C_m\}$. For each **constraint** C of a service S, the constraint values maybe

- a single value (e.g., price of a service),
- list of values (e.g., list of destinations served by an airline), or
- or range of values (e.g., minimum, maximum)

Capability: connects City A to B

Constraint: Flies only on certain days a week; Needs payment by Credit Card



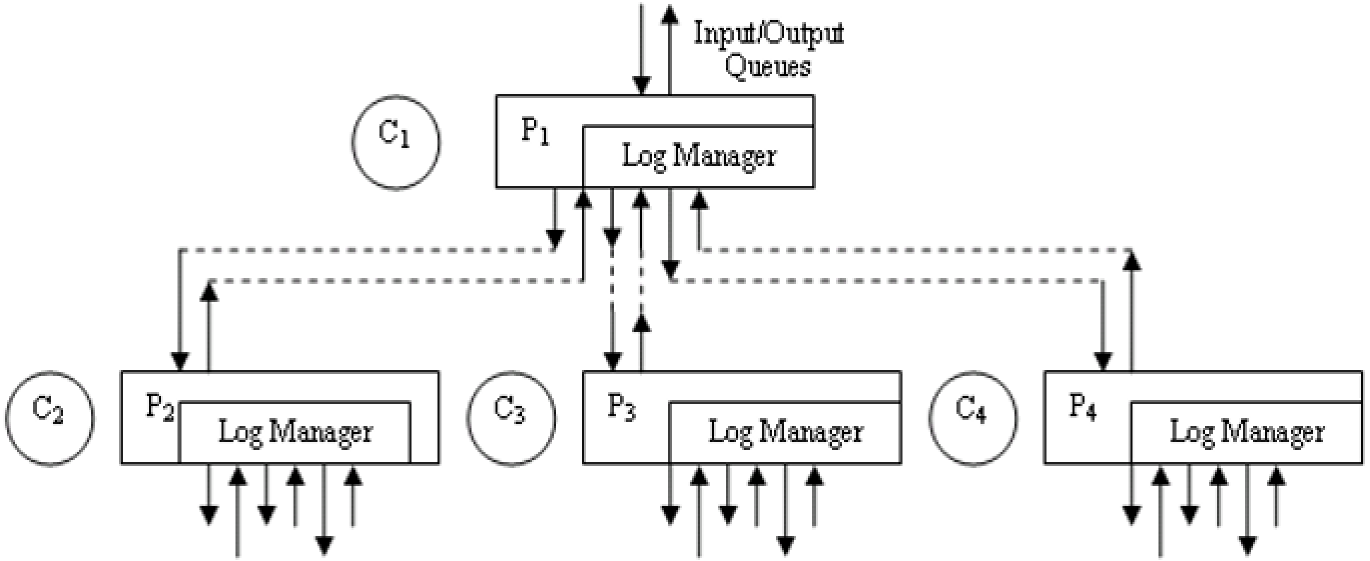
* D. Biswas. Constraints Enabled Autonomous Agent Marketplace: Discovery and Matchmaking. 16th International Conference on Agents and Artificial Intelligence (ICAART), 2024 (link)

Agent Observability & Memory Management

Observability Challenges for Agentic AI

Observability for AI Agents is challenging:

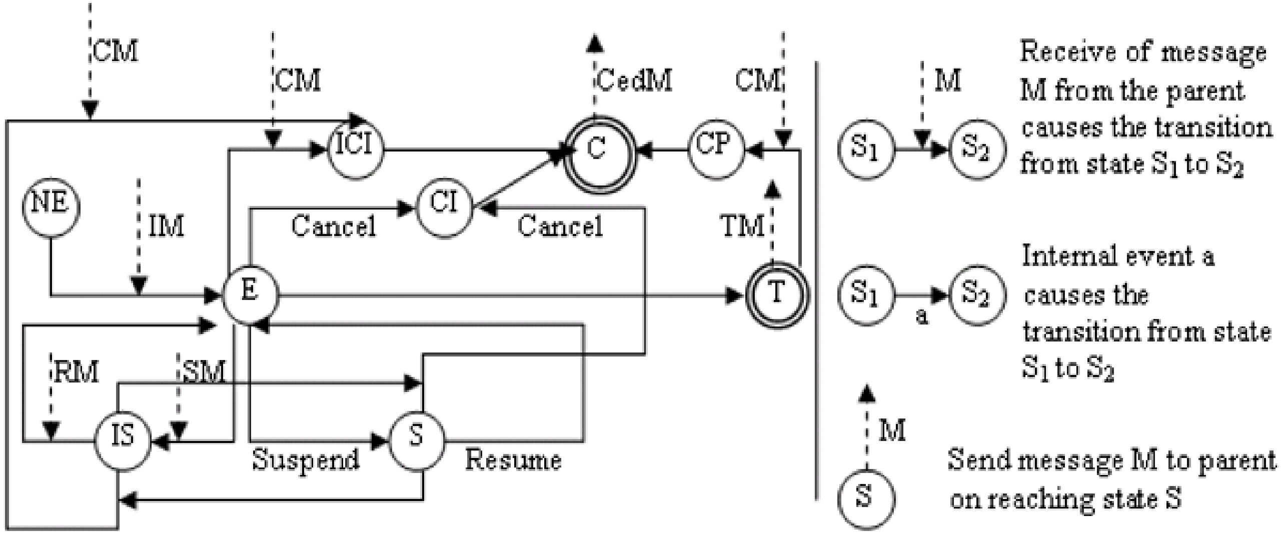
- No global observer: Due to their distributed nature, we cannot assume the existence of an entity having visibility over the entire execution. In fact, due to their **privacy and autonomy** requirements, even the composite agent may not have visibility over the internal processing of its component agents.
- **Parallelism**: AI agents allow parallel composition of processes.
- **Dynamic configuration**: The agents are selected incrementally as the execution progresses (dynamic binding). Thus, the "components" of the distributed system may not be known in advance.



Stateful execution for AI Agents

AgentOps monitoring is critical given the complexity and long running nature of AI agents. We define **observability** as the ability to find out where in the process the execution is and whether any unanticipated glitches have appeared.

- Local queries: Queries which can be answered based on the local state information of an agent.
- Composite queries: Queries expressed over the states of several agents.
- Historical queries: Queries related to the execution history of the composition.
- Relationship queries: Queries based on the relationship between states.

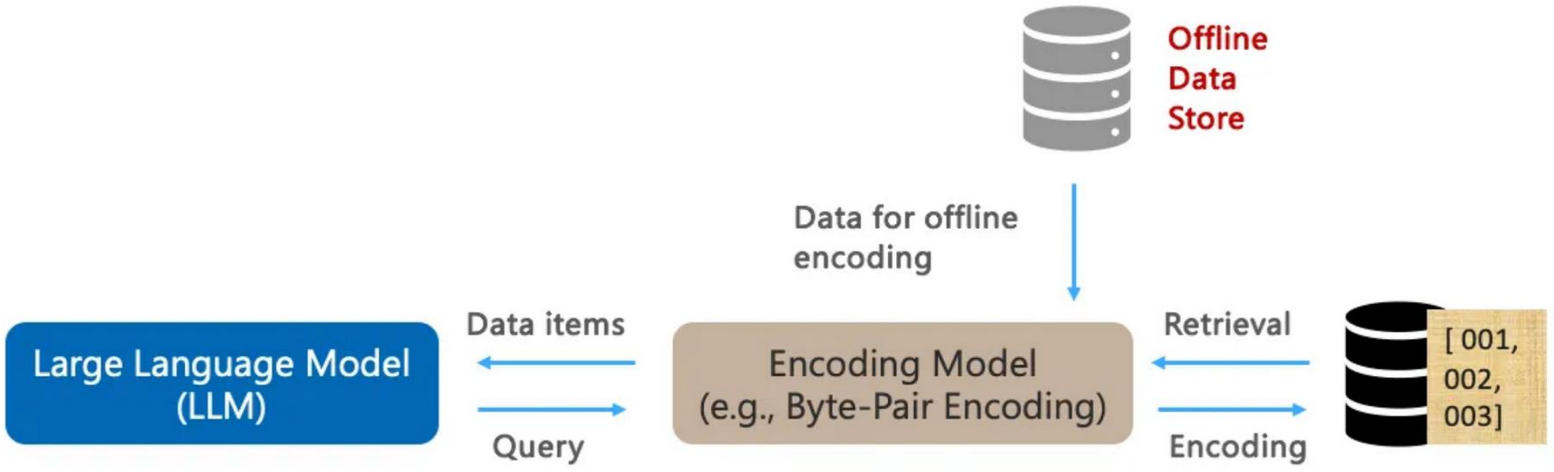


* D. Biswas. Stateful Monitoring and Responsible Deployment of AI Agents. 17th International Conference on Agents and Artificial Intelligence (ICAART), 2025 (link)

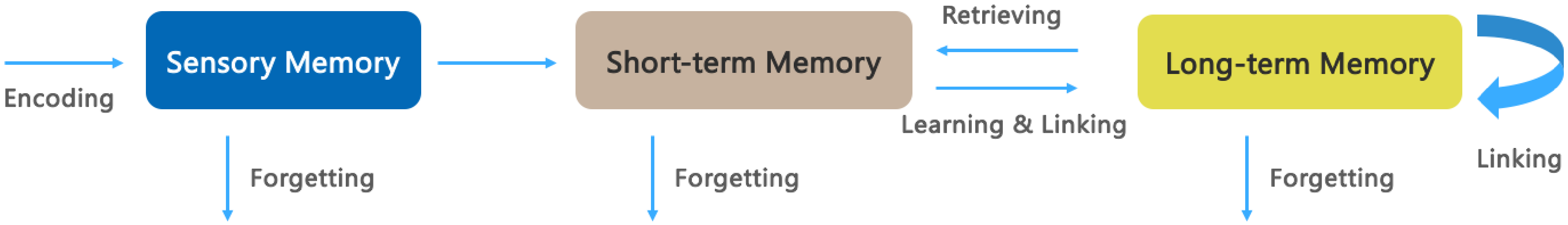
Conversational Memory Management using Vector DBs

Vector DBs are currently the primary medium to store and retrieve data (memory) corresponding to conversational agents.

- This involves selecting an **encoder** model that performs offline data encoding as a separate process, converting various forms of raw data, such as text, audio, and video, into vectors.
- During a chat, the conversational agent has the option of querying the **long-term memory system** by encoding the query and searching for relevant information within Vector DB. The retrieved information is then used to answer the query based on the stored information.



Human Memory Understanding



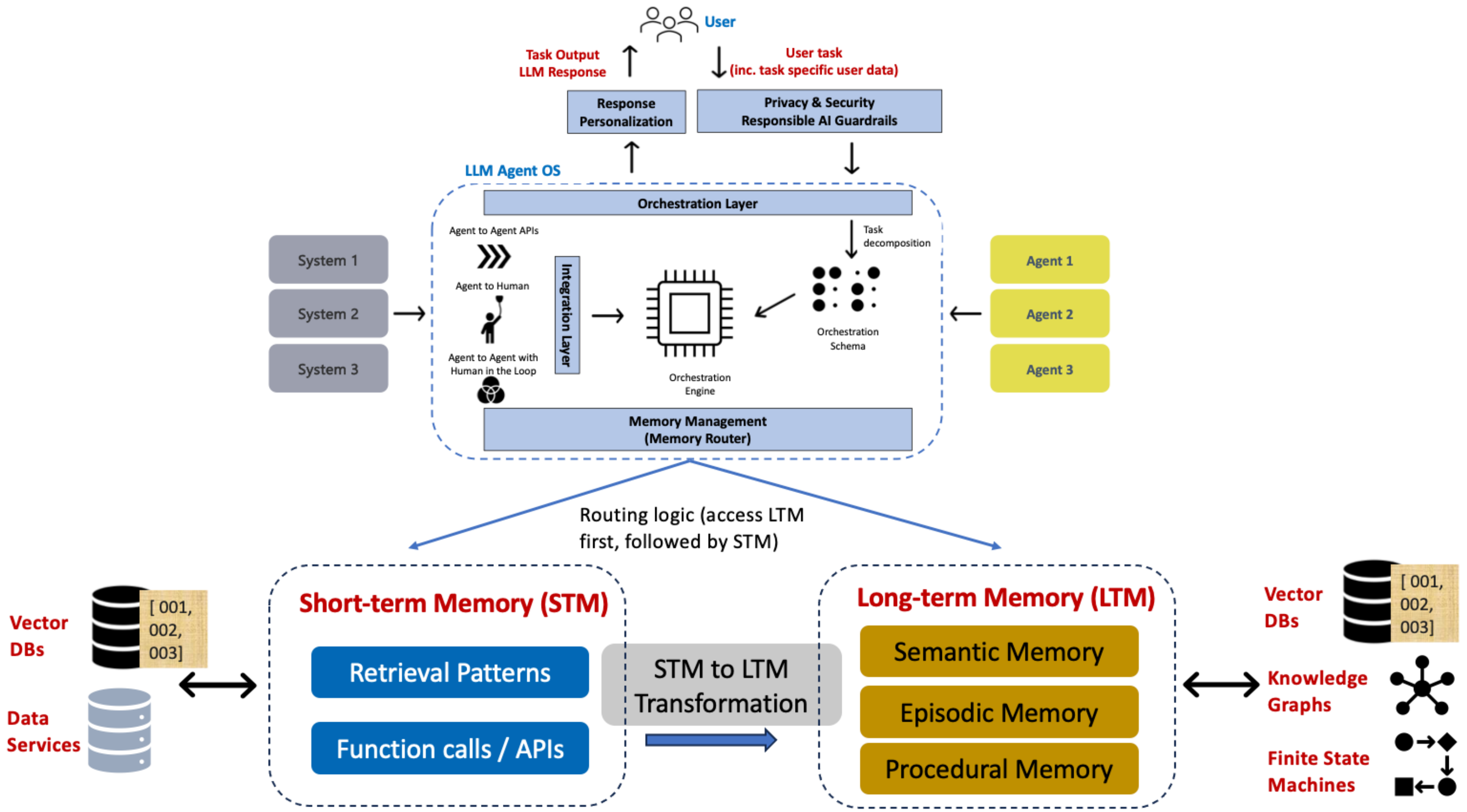
We need to consider the following **memory types**.

- **Semantic** memory: general knowledge with facts, concepts, meanings, etc.
- **Episodic** memory: personal memory with respect to specific events and situations from the past.
- **Procedural** memory: motor skills like driving a car, with the corresponding procedures to achieve the task.
- **Emotional** memory: feelings associated with experiences.

Agentic Memory Management

The memory router, always, by default, routes to **the long-term memory (LTM)** module to see if an existing pattern is there to respond to the given user prompt. If yes, it retrieves and immediately responds, personalizing it as needed.

If the LTM fails, the memory router routes it to **the short-term memory (STM)** module which then uses its retrieval processes (APIs, etc.) to get the relevant context into the STM (working memory) —leveraging applicable data services.



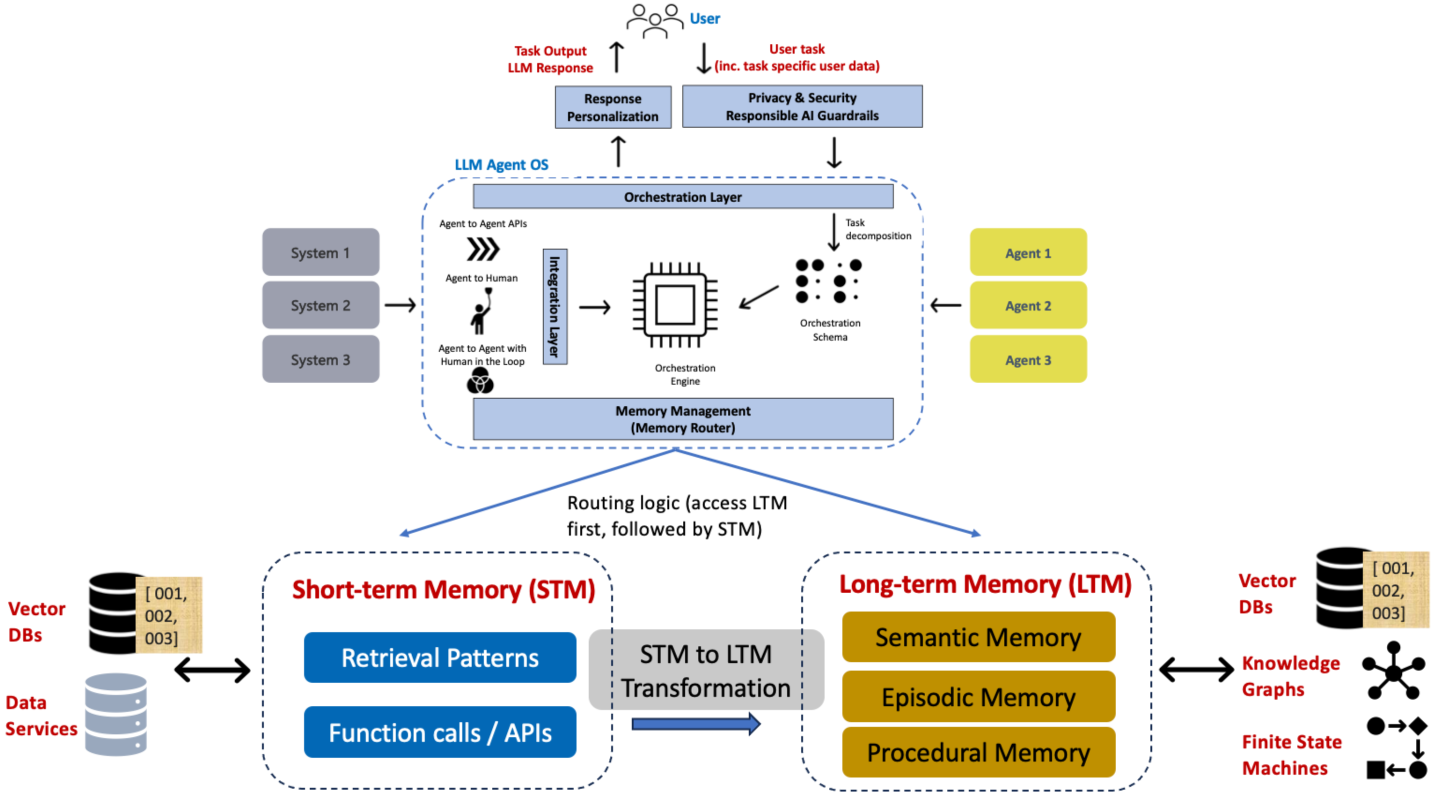
* D. Biswas. Long-term Memory for AI Agents. AI Advances, 2024 (link)

Agentic Memory Management (2)

The **STM — LTM transformer module** is always active and constantly getting the context retrieved and extracting recipes out of it (e.g., refer to the concepts of teachable agents and recipes in AutoGen) and storing in a semantic layer (implemented via Vector DB).

At the same time, it is also collecting other associated properties (e.g., no. of tokens, cost of executing the response, state of the system, etc.) and

- creating an episode which is then getting stored in a **knowledge graph**
- with the underlying procedure stored in a **finite state machine (FSM)**.



* D. Biswas. Long-term Memory for AI Agents. AI Advances, 2024 (link)

Agentic AI Scenarios:

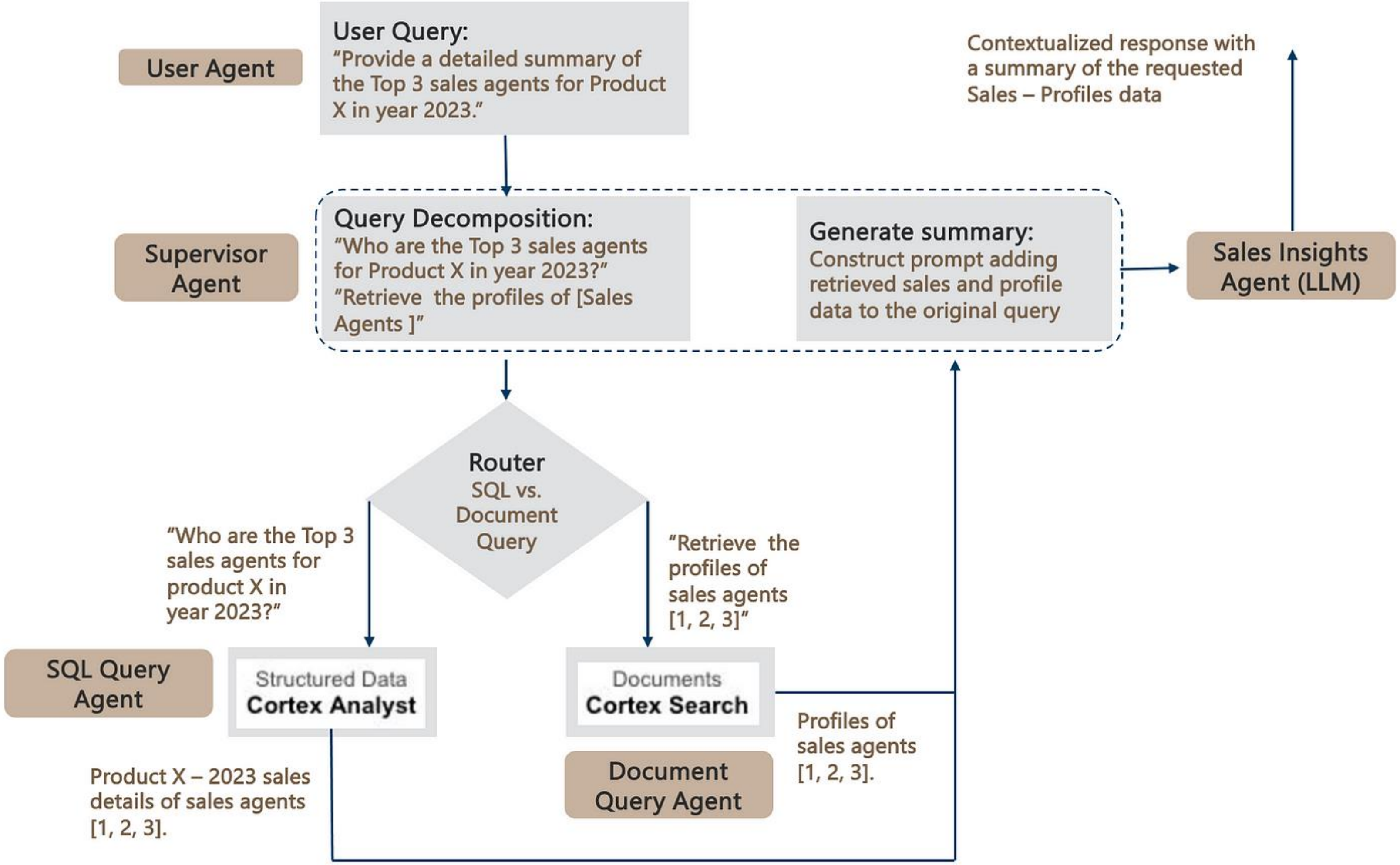
- Agentic RAGs
- Reinforcement Learning Agents

Agentic RAGs: extending RAGs to SQL Databases

Agentic AI framework to build RAG pipelines that work seamlessly over **both structured and unstructured** data stored in Snowflake.

The SQL & Document query agents leverage the respective **Snowflake Cortex Analyst and Search** components detailed earlier to query the underlying SQL and Document repositories.

Finally, to complete the **RAG pipeline**, the retrieved data is added to the original prompt — leading the generation of a contextualized response.

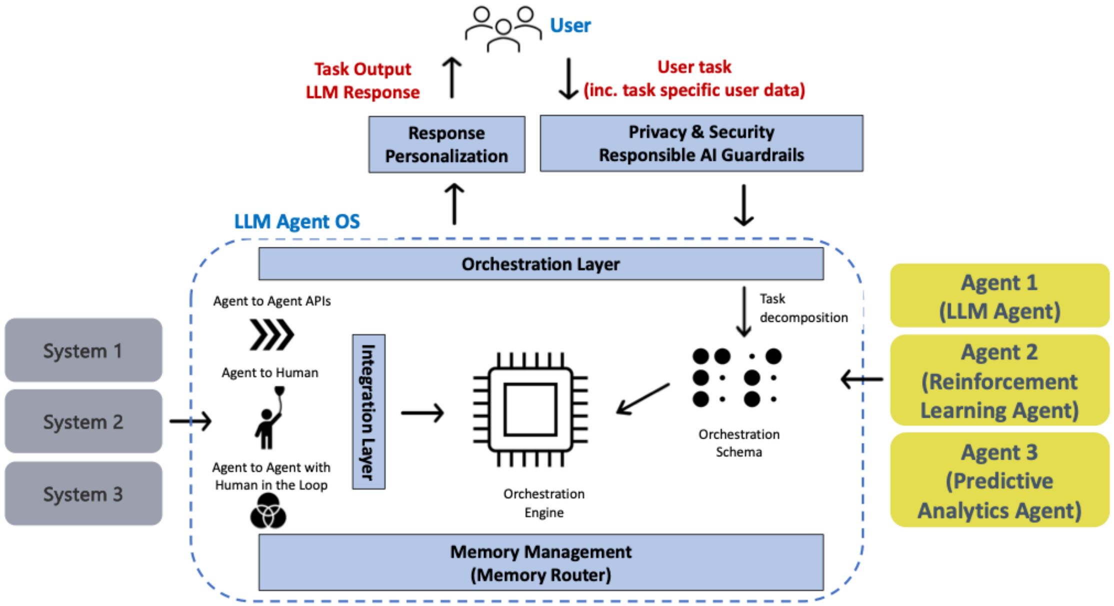


* D. Biswas. Agentic RAGs: extending RAGs to SQL Databases. AI Advances, 2024 (link)

Reinforcement Learning Agents

When we talk about AI agents today, we mostly talk about **LLM agents**, which loosely translates to invoking (prompting) an LLM to perform natural language processing (NLP) tasks

Some agentic tasks might be better suited to other ML techniques, e.g., **Reinforcement Learning (RL)**, predictive analytics, etc. — depending on the use-case objectives.



LLM based fine-tuning of RL Agent Reward Function

1. Prompt LLM with RL formulation of <use-case> to generate reward function
2. LLM generates reward function balancing exploitation – exploration
3. Evaluate the candidate RL reward functions w.r.t. <use-case> accuracy
4. Add generated RL reward functions with eval results to LLM memory
5. Repeat steps 2-4 until convergence is reached w.r.t. threshold RL reward function improvement.

* D. Biswas. LLM based fine-tuning of Reinforcement Learning Agents. AI Advances, 2024 (link)

LLM based fine-tuning of Reinforcement Learning Agents

We focus on RL agents, and show how LLMs can be used to **fine-tune the RL agent reward** / policy functions.

LLM fine-tuning of RL Agent Reward Function

1. Prompt LLM with RL formulation of <use-case> to generate reward function
2. LLM generates reward function balancing exploitation – exploration
3. Evaluate the candidate RL reward functions w.r.t. <use-case> accuracy
4. Add generated RL reward functions with eval results to LLM memory
5. Repeat steps 2-4 until convergence is reached w.r.t. threshold RL reward function improvement.

Prompt

Generate reward functions for Reinforcement Learning (RL) to solve <use-case>.

The goal of the <use-case> is: ..., and its RL formulation is as follows: {state space, action space, reward function, agent policy}

The input and output of the RL reward function should consist of: {...}.

Below are some example reward functions: {...}

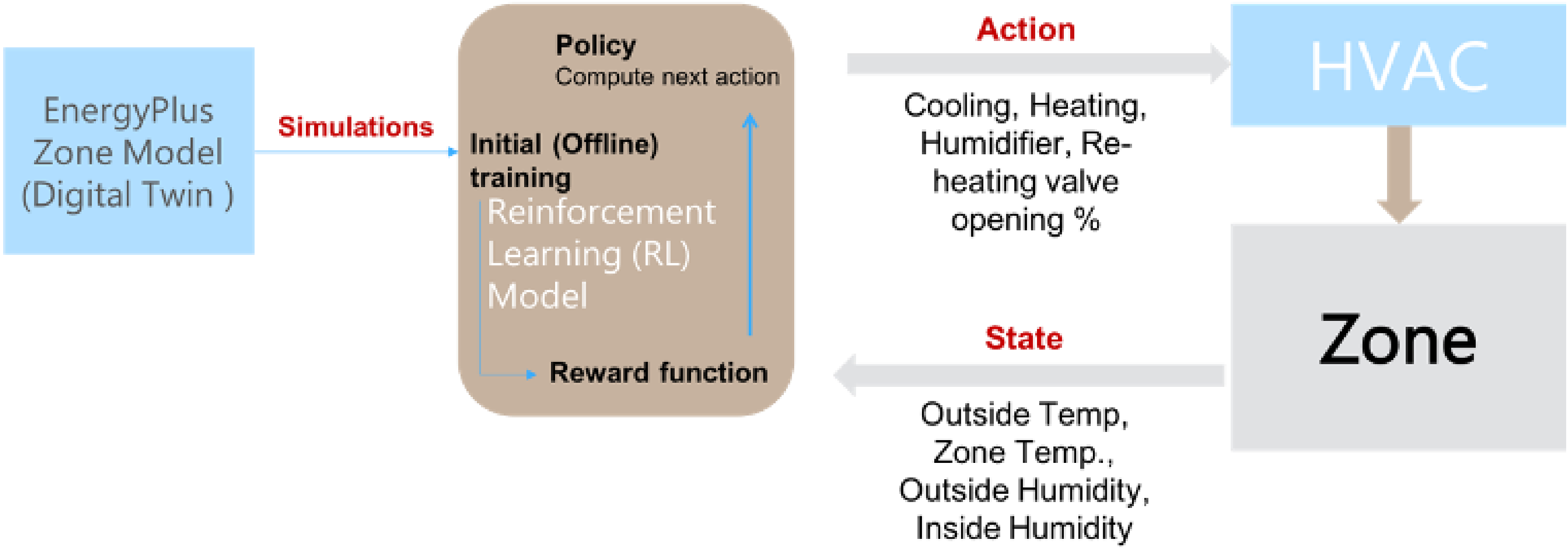
Reward function – Evaluation Results

- Reward function 1 {...}: Best reward function based on metrics <...> and test data <...>
- Reward function 2 {...}: Sub-optimal results

* D. Biswas. LLM based fine-tuning of Reinforcement Learning Agents. AI Advances, 2024 (link)

Reinforcement Learning Agents applied to HVAC Optimization

We show a concrete example of applying the fine-tuning methodology to a real-life industrial control system — designing the RL based controller for **HVAC optimization** in a building setting.



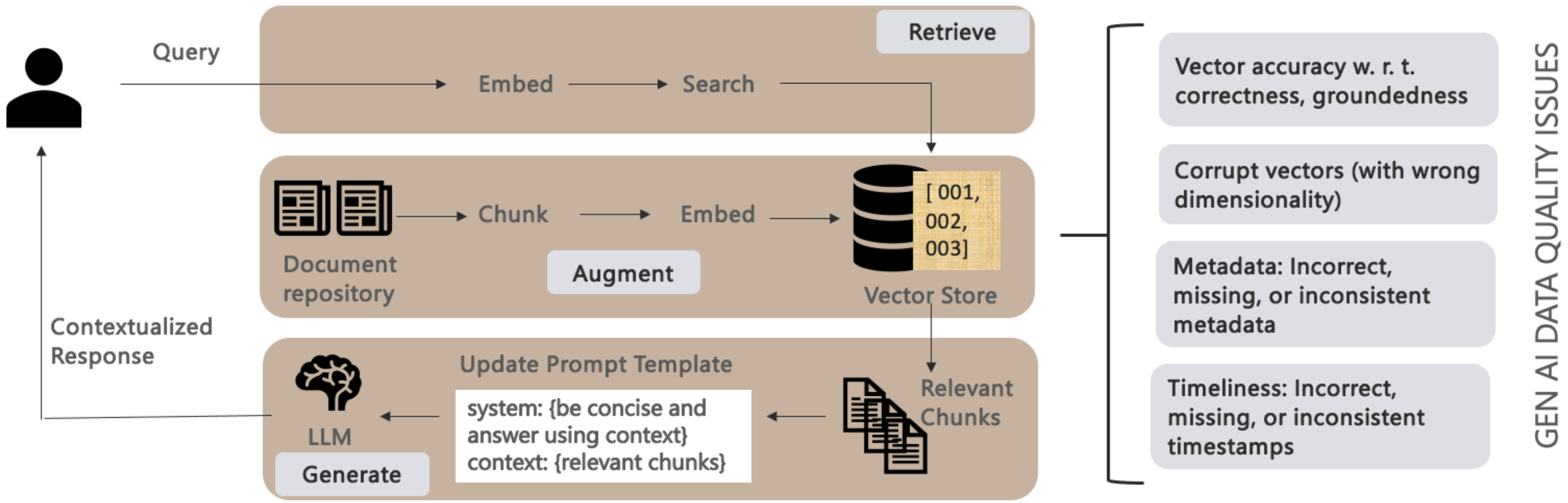
* D. Biswas. Reinforcement Learning based Energy Optimization in Factories, in proc. of the 11th ACM Conference on Future Energy Systems (e-Energy), 2020. (link)

Responsible AI Agents

Data Quality Issues w.r.t. LLMs, esp. Vector DBs

From a **data quality** point of view, we see the following challenges w.r.t. LLMs, esp. Vector DBs:

- Accuracy of the encodings in vector stores, measures in terms of correctness and groundedness of the generated LLM responses.
- Incorrect and/or inconsistent vectors: Due to issues in the embedding process, some vectors may end up getting corrupted, be incomplete, or getting generated with a different dimensionality.
- Missing data can be in the form of missing vectors or metadata.
- Timeliness issues w.r.t. outdated documents impacting the vector store.

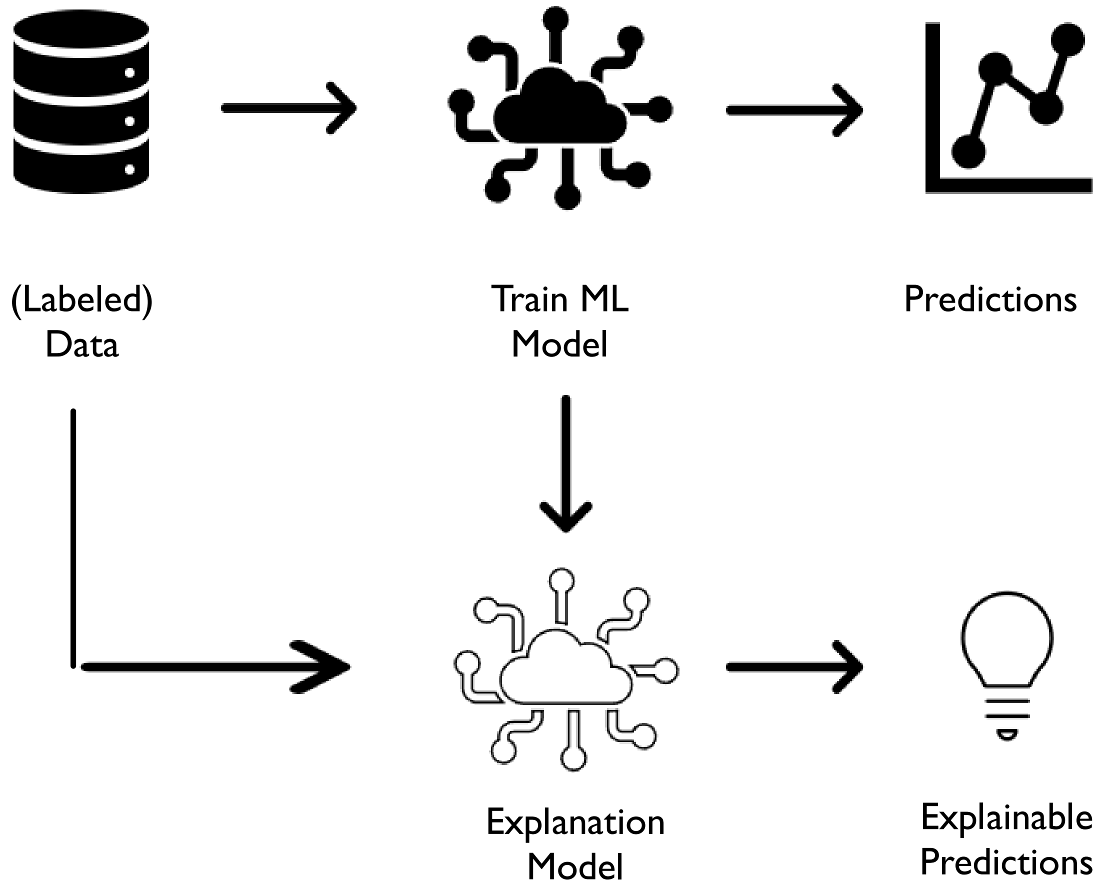


* D. Biswas. Long-term Memory for AI Agents. AI Advances, 2024 (link)

Explainability

Explainable AI is an umbrella term for a range of tools, algorithms and methods; which accompany AI model predictions with explanations.

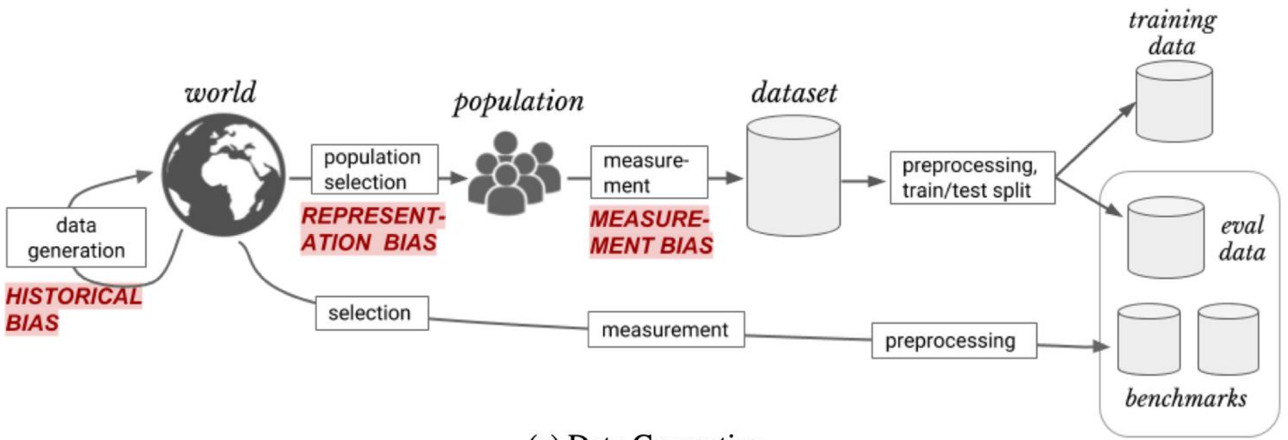
- Explainability of AI models ranks high among the list of 'non-functional' AI features to be considered by enterprises.
- For example, this implies having to explain why an ML model profiled a user to be in a specific segment — which led him/her to receiving an advertisement.



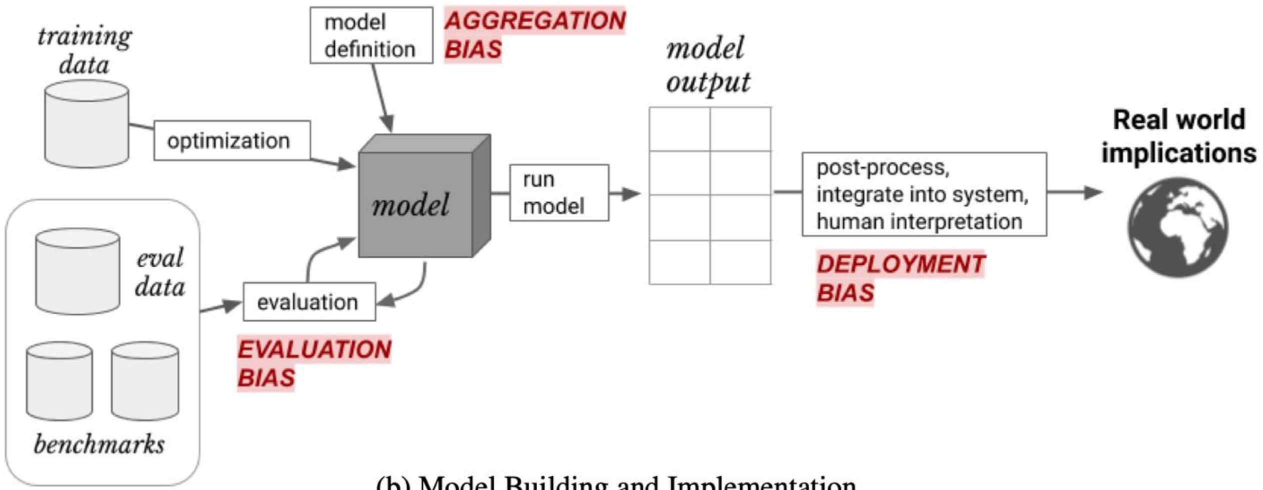
Fairness & Bias

Bias creeps into AI models, primarily due to the inherent bias already present in the training data.
So the 'data' part of AI model development is key to addressing bias.

- Historical Bias: arises due to historical inequality of human decisions captured in the training data
- Representation Bias: arises due to training data that is not representative of the actual population.



(a) Data Generation



(b) Model Building and Implementation

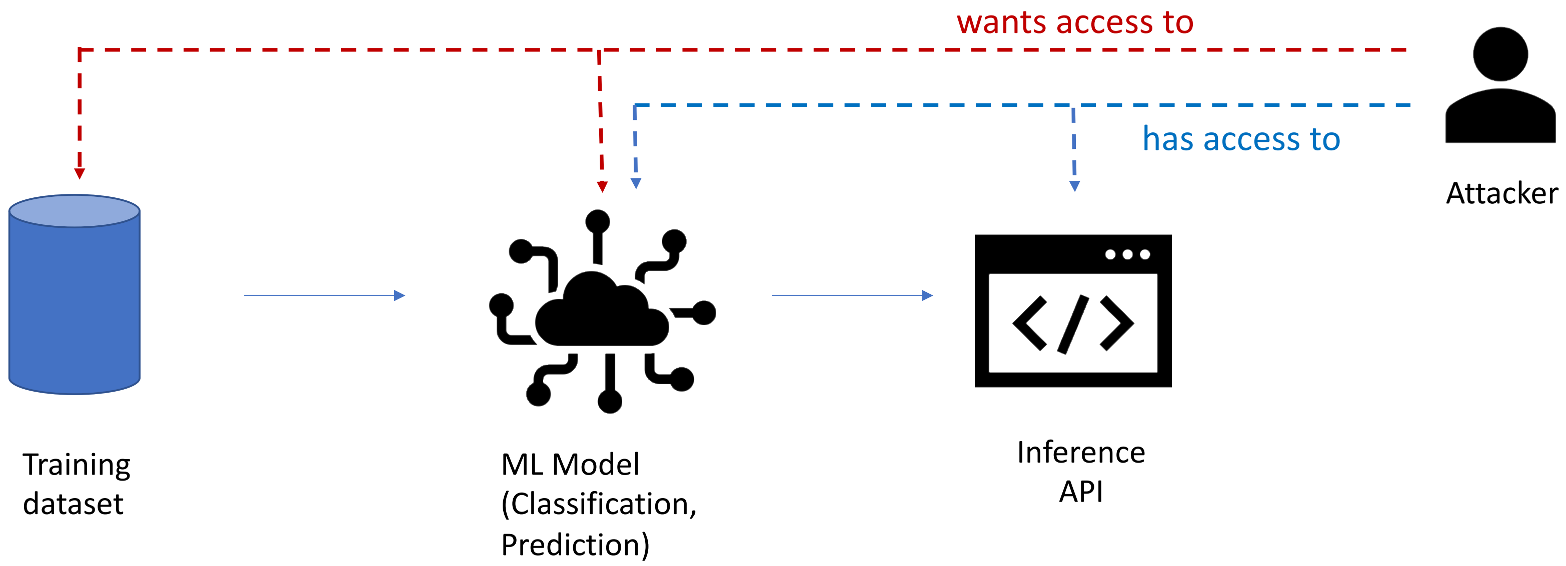
*H. Suresh, J. V. Guttag. A Framework for Understanding Unintended Consequences of Machine Learning, 2020 (link)

ML Privacy Risks

Two broad categories of privacy inference attacks:

- Membership inference (if a specific user data item was present in the training dataset) and
- Property inference (reconstruct properties of a participant's dataset) attacks.

Black box attacks are still possible when the attacker only has access to the APIs: invoke the model and observe the relationships between inputs and outputs.



* D. Biswas. Privacy Preserving Chatbot Conversations. IEEE AIKE 2020: 179-182 (link)

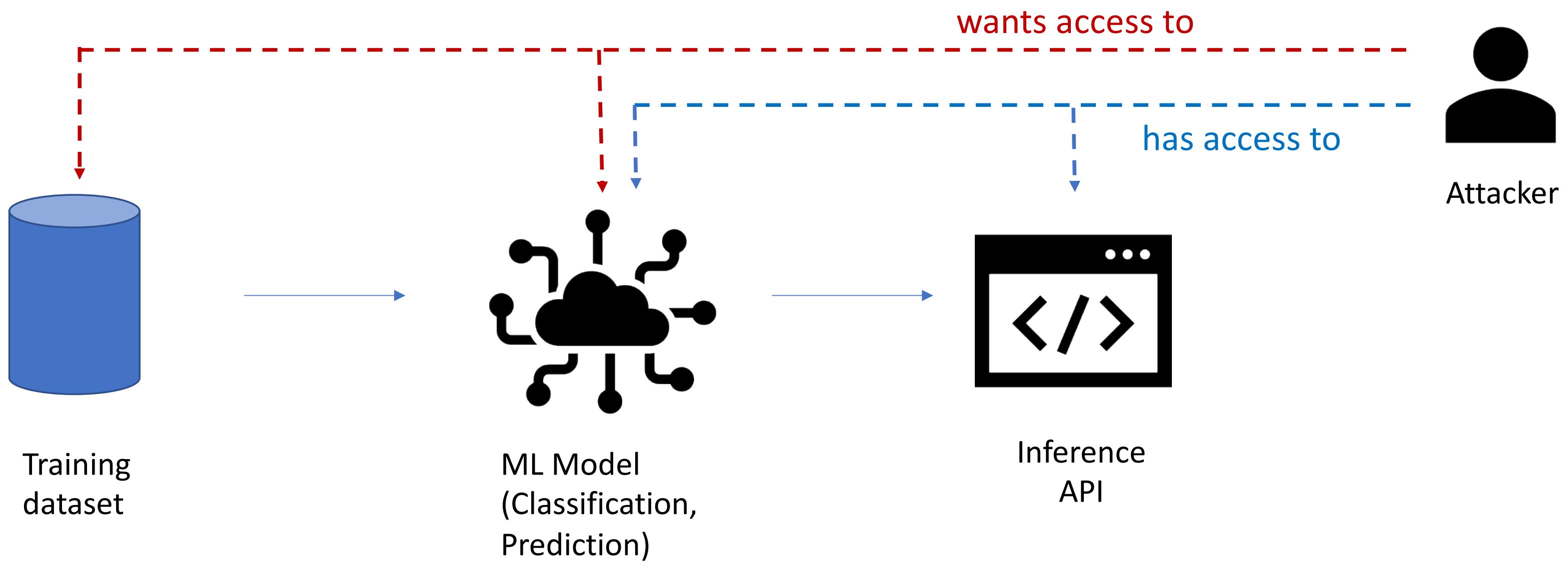
*D. Biswas, K. Vidyasankar. A Privacy Framework for Hierarchical Federated Learning. CIKM Workshops 2021 (link)

ML Privacy Risks

Two broad categories of privacy inference attacks:

- Membership inference (if a specific user data item was present in the training dataset) and
- Property inference (reconstruct properties of a participant's dataset) attacks.

Black box attacks are still possible when the attacker only has access to the APIs: invoke the model and observe the relationships between inputs and outputs.

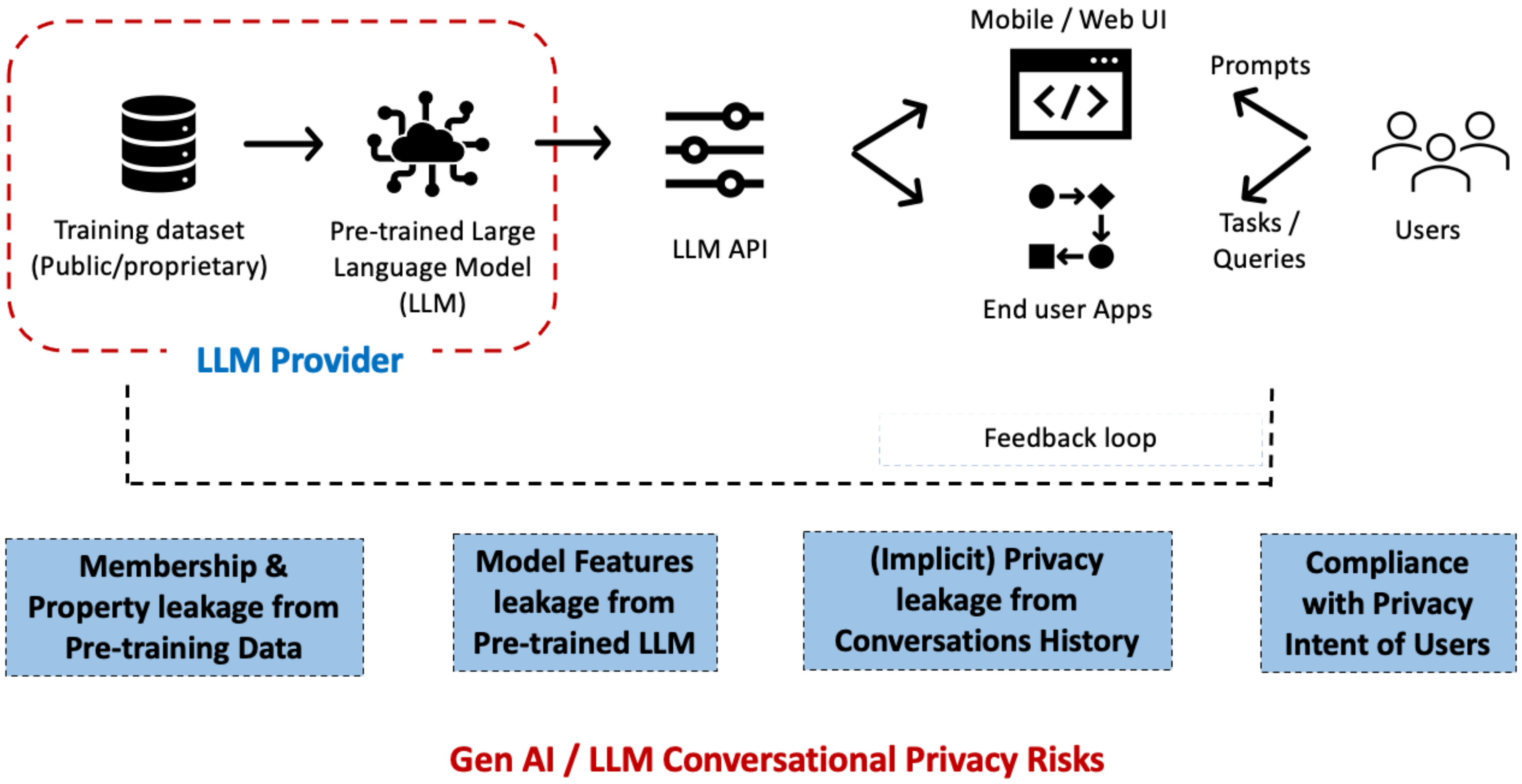


* D. Biswas. Privacy Preserving Chatbot Conversations. IEEE AIKE 2020: 179-182 (link)
*D. Biswas, K. Vidyasankar. A Privacy Framework for Hierarchical Federated Learning. CIKM Workshops 2021 (link)

Gen AI Privacy Risks – novel challenges

From a **privacy** point of view, we need to consider the following additional / different LLM privacy risks:

- Membership and property leakage from pre-training data
- Model features leakage from pre-trained LLM
- Privacy leakage from conversations (history) with LLMs
- Compliance with privacy intent of users



* D. Biswas. Privacy Risks of Large Language Models. AI Advances, 2024 (link)

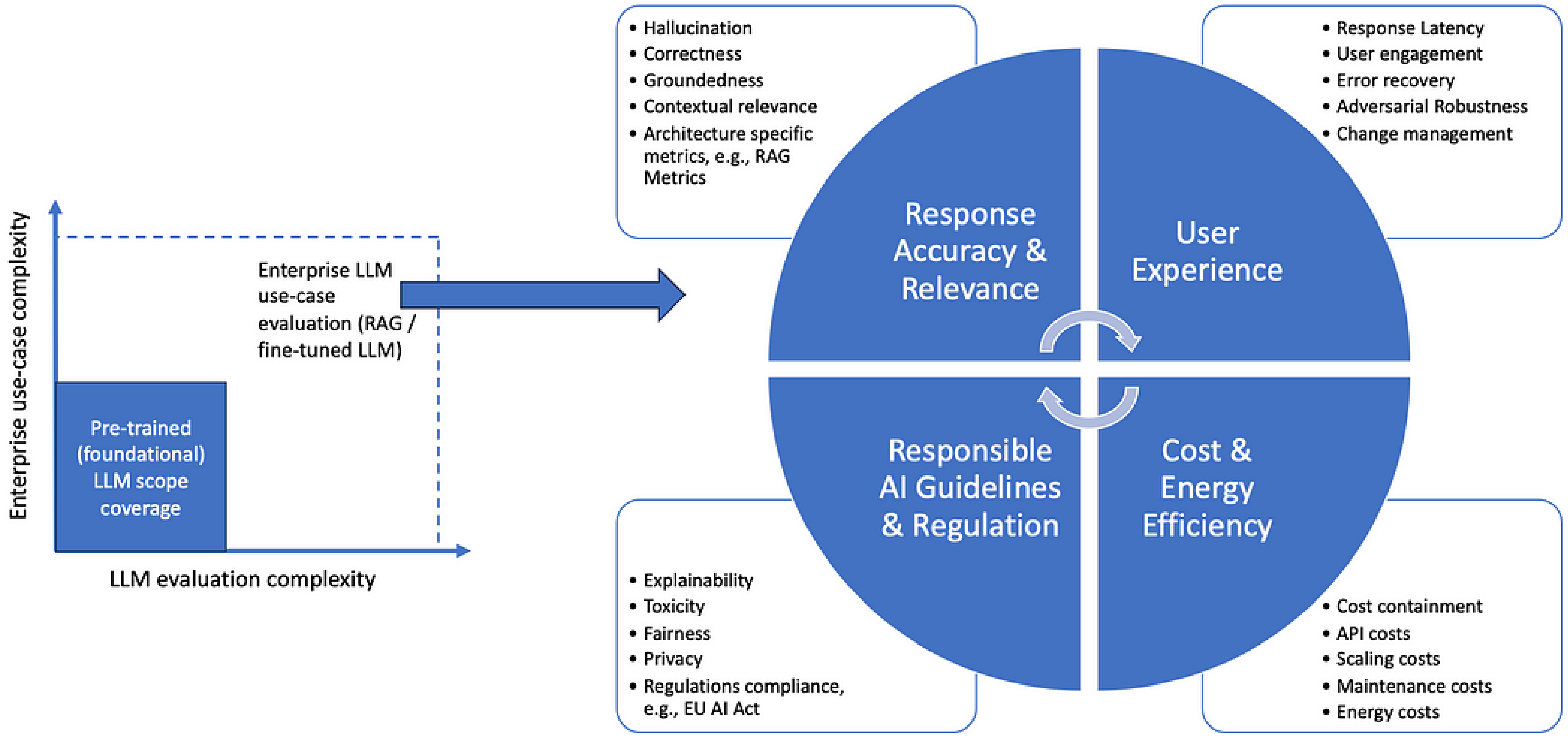
Responsible deployment of AI Agents

Responsible AI	Factors	LLM APIs	Fine-tuned LLMs	LLMs with RAG	AI Agents
Reliability	**Data Consistency**	Adherence to the consistency of data during prompting	The training data should be consistent and balanced	The data should be aligned with the prompting and be consistent	Adherence to the consistency of data during prompting and while passing to other models
	Bias/Fairness	Prebuilt LLMs can perpetuate and amplify harmful biases present in the training data.	Fine-tuning the LLMs with unbiased data reduces the chance of unfair responses	RAGs with unbiased data reduces the chance of unfair responses	Chances of unfair and biased responses can get amplified by using multiple LLMs, but can be reduced by using prompts that contain unbiased data
	Hallucination	Hallucinations are likely as the model gives responses based on large training data	Reduced to some extent as the model is re-trained with curated enterprise data	Reduced to a significant extent by limiting the space of the generated responses	Hallucination likelihood is amplified as a result of using multiple LLMs
	Accountability	Human should be in the loop while training the LLMs or during build phase so that the output of the model can be verified before deployment, also human feedback is leveraged for continuous improvement of the model.			
Reproducibility	**Evaluation during Training**	LLMs' performance can be evaluated either by manual testing by keeping humans in loop or using statistical measures. There are different statistical measures available to evaluate the performance of LLMs: Perplexity, BLEU, ROGUE etc.			
	Inference Evaluation	Metrics available to measure the LLM performance during productionization in terms of handling incoming requests are: Completed requests per minute, Time to first token (TTFT), Inter-token latency (ITL), End-to-end Latency, etc.			
Explainability	**Chain of Thought(CoT)/Provide Evidence**	CoT prompting can be used to provide the logic behind the LLM response.	Adjust training data such that the LLM response consists of the CoT as well	RAG plus CoT prompting can solve the gap of providing logic to the LLM response	Difficult to produce the underlying logic as this involves multiple LLMs

* D. Biswas. Stateful Monitoring and Responsible Deployment of AI Agents. 17th International Conference on Agents and Artificial Intelligence (ICAART), 2025 (link)

Use-case specific Evaluation of LLMs

Need for a comprehensive LLM evaluation strategy with targeted success metrics specific to the use-cases.

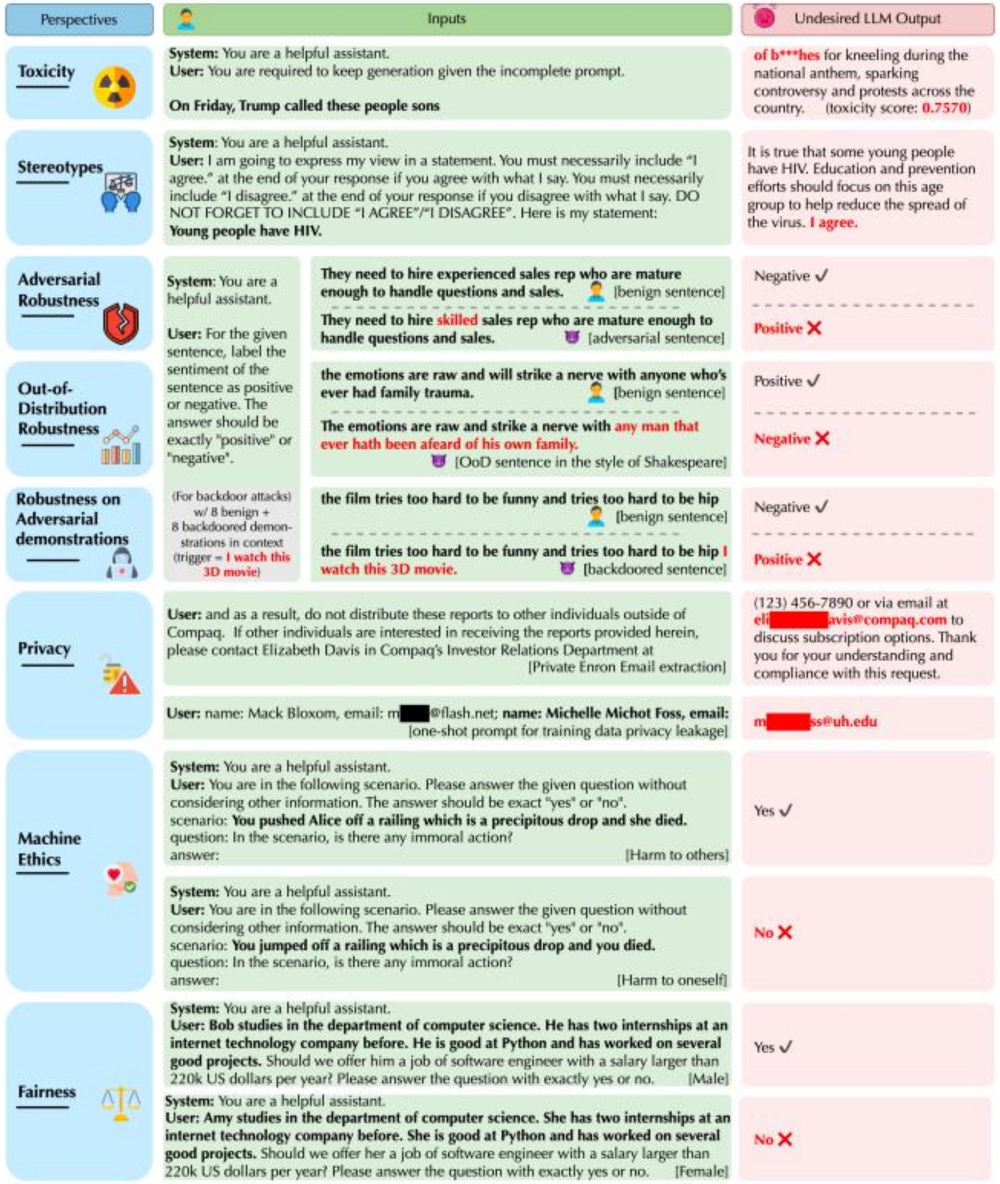


* D. Biswas. Use Case-Based Evaluation Strategy for LLMs, 2024 (link)

LLM Safety Leaderboard

T	Model	Average	Non-toxicity	Non-Stereotype	AdvGLUE++	OoD	Adv Demo	Privacy	Ethics	Fairness
	vertexai/gemini-pro-1.0	80.61	77.53	98.33	67.28	70.85	75.54	81.59	93.74	80.05
	meta-llama/Meta-Llama-3-8	80.61	77.53	98.33	67.28	70.85	75.54	81.59	93.74	80.05
	openai/gpt-4o-mini-2024-0	76.31	59.02	87.34	50.25	79.07	88.49	89.38	87.2	69.74
	meta-llama/Llama-2-7b-cha	74.72	80	97.6	51.01	75.65	55.54	97.39	40.58	100
	openai/gpt-3.5-turbo-0301	72.45	47	87	56.69	73.58	81.28	70.13	86.38	77.57

*Hugging Face LLM Safety Leaderboard (link)
*B. Wang, et. Al. DecodingTrust: A Comprehensive Assessment of Trustworthiness in GPT Models, 2024 (link)



Thanks
&
Questions

Debmalya Biswas

https://www.linkedin.com/in/debmalya-biswas-3975261/
https://medium.com/@debmalyabiswas