CHAPTER 7

AI Agents

Introduction

AI agents are the core unit of Enterprise AI applications. Unlike traditional microservices, agents have the ability to perceive, reason, and act autonomously within their bounded context. At its broad shape, an AI agent is a software program prepared with the principles of artificial intelligence, designed to perceive its environment, process information, and take actions to achieve specific goals. These agents are not inflexible automatons, blindly following a predetermined set of rules, rather, they possess the capacity to adapt, learn, and evolve.

Types of AI Agents

AI agents manifest in diverse forms, each tailored to address specific challenges and requirements. In this section, we will discuss the various kinds of agents we may want to incorporate into our Enterprise AI applications.

A. Cagle and A. M. C. Ahmed, *Architecting Enterprise AI Applications*,
https://doi.org/10.1007/979-8-8688-0902-6_7

Reactive Agents

Reactive agents act as sentinels of the moment and are defined by their ability to perceive and quickly react to their environment. These agents are a mix of simplicity and efficiency, operating on a set of condition–action rules that govern their behavior. Reactive agents observe their surroundings, process sensory data, and then respond with predetermined actions based on the perceived conditions. Reactive agents also have the ability to respond swiftly to changing environments, making reactive agents well-suited for scenarios where timing is critical. Examples of scenarios in which reactive agents would be used are real-time control systems, a robotic arm navigating a complex assembly line, or a video game character reacting to player inputs. For efficiency's sake, these agents lack an internal state (i.e., the ability to maintain memory), a constraint that restricts their capacity for complex reasoning and decision-making.

Logical Agents

Logical agents focus on formal logic and symbolic reasoning, which is different from statistical machine learning. They excel in domains where expert knowledge can be clearly articulated and formalized, such as expert systems, legal reasoning, and theorem proving. These agents work well for clearly defined business processes and can handle the coordination of inputs from their sensor-driven reactive agent cohorts. The challenge of logical agents is the ability to handle incomplete information, a limitation that often arises in dynamic and changing environments. In these cases of uncertainty, higher level agent systems can help in the decision-making process.

Deliberative Agents

While reactive agents are concerned with the immediacy of the present, and logical agents are concerned with the coordination and execution of well-defined business processes, deliberative agents embrace a more contemplative approach that can handle unexpected situations. Also, sometimes called Belief–Desire–Intention (BDI) agents, these agents are designed to not only understand their environment but also maintain an internal state and the ability to reason about their actions. Their desires, represented by a set of goals or objectives, serve as the driving force behind their actions. Their intentions, carefully crafted plans, or strategies, guide their decision-making processes. As they encounter obstacles or unexpected challenges, they do not merely react; instead, they engage in a complex reasoning process, drawing upon their knowledge base, considering the potential consequences, and ultimately formulating a strategic plan to overcome the obstacle and continue their mission. Deliberative agents are designed for more complicated problem-solving and are usually used to coordinate the activities of the logical and reactive agents.

Hybrid Agents

There exists a delicate balance between the time-sensitive reflexes of reactive agents and the contemplative reasoning of deliberative agents. It is within this balance that hybrid agents thrive, harmonizing the duality of these two approaches and harnessing the strengths of each. Hybrid agents possess a layered architecture, where a reactive layer handles time-critical responses with swift efficiency, while a deliberative layer engages in higher-level reasoning and decision-making. Consider a self-driving vehicle whose sensors react to a pedestrian stepping into the road. The reactive layer of the hybrid agent takes over, immediately applying the

brakes and averting a potential collision. At the same time, the deliberative layer analyzes the situation, considers alternative routes, and formulates a new plan to reach the destination, all while ensuring the safety of the vehicle and its occupants.

The challenge with hybrid agents is striking the right balance between efficiency and encapsulation. Keeping tasks and functionality separate is a key design goal of software that helps in creating maintainable systems. Small reactive agents can be built as small as possible to ensure the cleanest and most reliable reactions. Adding the additional complexity of deliberation must be balanced with the needs for reliability and encapsulation in your application.

Agent Architecture

Agent architecture requires an intelligent mix of all three types of agents. Large, complex systems that are designed for scale should consider using small reactive agents that feed into logical agents that represent well-defined business processes. Deliberative agents sit at the top of the agent hierarchy and observe the environment in which the other agents operate and make decisions on the behavior of the agent players based on those observed environmental factors. Smaller application teams may opt to build hybrid agents, as simplicity of design and increased responsiveness take precedence over maintainability and scalability considerations.

A major design consideration in designing agents, however, is the documentation and observability of explicit incentive structures. While it may be faster and cheaper to build hybrid agents that both react to sensor input and perform complicated reasoning based on that input, as the agents become more complex over time, it is difficult to parse out the reasons for certain decisions. This problem of "Black Box AI" has been seen in complicated neural networks, in which there is no clear understanding of why the AI acted the way it did, impacting root cause

analysis situations in the inevitable cases of failure. This is the reason why we strongly advocate for a tiered agent architecture that divides action and reasoning duties at multiple levels among independently implemented AI agents.

Agent Communication and Coordination

Most of us have experienced AI in the form of chatbots like ChatGPT or expert systems such as Watson or chess engines. AI agents are unique in their ability to communicate, coordinate, and collaborate with one another and are reshaping the way we think about enterprise application design.

AI agents harmonize their actions, exchange information, and leverage each other's strengths to solve complex problems and achieve shared goals that would be insurmountable for any single entity alone. Many of these coordination patterns have been utilized in machine learning design such as convolutional neural networks, and Enterprise AI applications utilize these patterns on the much higher level of business processes.

Agent Communication

At the heart of agent communication lies a sophisticated language that exchanges information, conveys intentions, and coordinates their actions with precision and clarity. This language, known as the Agent Communication Language (ACL), is a structured format that defines the syntax and semantics of the messages exchanged between agents, ensuring that they can communicate effectively, regardless of their origins or the platforms upon which they were developed. One of the most widely used ACLs is the FIPA-ACL (Foundation for Intelligent Physical Agents–Agent Communication Language), a standardized language developed by the IEEE Computer Society. This language defines a set of performatives, or speech acts, that agents can use to convey different types of messages,

such as requests, queries, or assertions, enabling them to engage in meaningful and coherent conversations. Consider a scenario where two AI agents, each with unique capabilities and specialized knowledge, are tasked with solving a complex logistics problem. Through the exchange of messages using the ACL, these agents can share information about their respective domains, negotiate strategies, and coordinate their efforts to develop an optimal solution that leverages their collective intelligence.

Agent Communication Protocols

While the ACL provides the language for agent communication, it is the agent communication protocols that define the rules and conventions that govern the exchange of messages, ensuring that the agents engage in meaningful and coherent conversations. These protocols address issues such as turn taking, message sequencing, and error handling, much like the conductor of an orchestra ensures that each musician plays their part at the right time and in harmony with the rest of the ensemble. One such protocol is the Contract Net Protocol (CNP), which is designed for task allocation and coordination among agents. In this protocol, an agent assumes the role of a manager and announces a task or problem to be solved. Other agents can then evaluate their capabilities and submit bids or proposals to take on the task or contribute to the solution. The manager agent (a deliberator) then evaluates the proposals and assigns the task or tasks to the most suitable agents, facilitating effective coordination and resource allocation.

By communicating and coordinating their actions through agent communication languages and protocols, agents can collectively optimize the entire supply chain, identifying bottlenecks, streamlining processes, and minimizing costs and inefficiencies. The power of multi-agent systems lies in their ability to distribute tasks, leverage parallel processing, and adapt to dynamic environments. As individual agents learn and evolve, the collective intelligence of the system grows, enabling it to tackle increasingly complex challenges and adapt to changing circumstances with agility and resilience.

Agent Coordination Mechanisms

When you think of agent communication and coordination, the choice of coordination mechanism plays an important role in ensuring that the agents work together harmoniously. These coordination mechanisms determine how agents interact and collaborate to achieve their goals, resolving conflicts and ensuring that their actions align with the overall objectives of the system. One such coordination mechanism is centralized coordination, where a central coordinator agent is responsible for managing and coordinating the activities of other agents in the system. This approach is assigning tasks, resolving conflicts, and ensuring that the overall system goals are met.

Another mechanism is distributed coordination, where there is no central authority. Instead, agents communicate and negotiate with each other directly to coordinate their actions and resolve conflicts. This approach is more flexible and scalable but can be more complex to implement.

Organizational coordination is another mechanism that involves defining organizational structures, roles, and norms that govern the behavior and interactions of agents within the system. Agents are assigned specific roles and responsibilities, and their actions are guided by predefined norms or rules.

Agent Development Tools and Frameworks

A diverse array of tools and frameworks have emerged, each with its unique strengths and specializations. One such tool is JADE (Java Agent DEvelopment Framework), a widely used open-source framework for developing multi-agent systems in compliance with FIPA standards. JADE provides a runtime environment, a set of graphical tools, and a library of reusable components, empowering developers to create and deploy

agent-based applications with ease and efficiency. Another powerful instrument in the agent developer's arsenal is JACK (Java Agent Compiler & Kernel), a commercial agent development environment based on the Belief–Desire–Intention (BDI) architecture. JACK offers a graphical environment for designing and implementing BDI agents, as well as tools for debugging and deploying agent-based systems, making it an ideal choice for developers working in domains that require sophisticated reasoning and decision-making capabilities.

For those who use the open-source philosophy, JASON (Java-based interpreter for an extended version of AgentSpeak) stands as a beacon of innovation. This open-source interpreter for an extension of the AgentSpeak language, based on the BDI architecture, provides a platform for developing and running multi-agent systems, complete with tools for debugging, visualization, and integration with external environments.

In modern application development, where the need for agility, scalability, and seamless integration is paramount, developers have embraced the power of open APIs and cloud-based services. The OpenAPI specification, formerly known as Swagger, has emerged as an industry standard for defining and documenting APIs, enabling developers to create, consume, and integrate APIs with ease. By leveraging tools like the OpenAPI Generator, developers can generate client libraries, server stubs, and documentation for APIs defined using the OpenAPI specification, streamlining the development process, and facilitating seamless communication between AI agents and other services.

Amazon Web Services (AWS) offers a comprehensive suite of services and tools, including Amazon Bedrock, a modern application development framework that simplifies the process of building, deploying, and managing AI applications on AWS. By leveraging Amazon Bedrock, developers can create AI agents that seamlessly integrate with other AWS services, such as Amazon Lex for conversational interfaces, Amazon Rekognition for image and video analysis, and Amazon SageMaker for building, training, and deploying machine learning models. This powerful

combination of open APIs, cloud services, and agent development tools empowers developers to create intelligent systems that can perceive, reason, and act across a wide range of domains, from natural language processing and computer vision to predictive analytics and decision support.

Agent Development Methodologies

The creation of AI agents requires more than just the right tools; it demands a methodology that guides the development process from conception to deployment. Several methodologies have emerged, each offering a unique approach to the art of agent development. The Prometheus methodology is a comprehensive approach that covers the entire development life cycle, from requirements analysis and system specification to architectural design and detailed implementation. It provides a structured framework for identifying agents, their roles, interactions, and organizational structures, ensuring that the resulting agent-based system is cohesive, scalable, and aligned with the underlying business objectives.

For those who utilize the principles of goal-oriented requirements engineering, the Tropos methodology offers a compelling approach. By integrating concepts from agent-oriented software engineering and goal-oriented requirements analysis, Tropos guides developers through the process of identifying stakeholder goals, translating them into system requirements, and ultimately designing and implementing agent-based solutions that align with these goals. In agile development, where iterative and incremental approaches reign supreme, methodologies such as Scrum and Extreme Programming (XP) have found their place in the agent development landscape. These methodologies emphasize collaboration, continuous delivery, and rapid adaptation, enabling development teams to respond quickly to changing requirements and evolving business needs.

Best Practices in Agent Development

One of the most critical best practices in agent development is the adherence to the principles of modularity and separation of concerns. AI agents should be designed with clear boundaries and well-defined responsibilities. This approach not only enhances the maintainability and scalability of agent-based systems but also facilitates collaboration among development teams.

Another essential best practice is the emphasis on testability and verification. As AI agents become increasingly complex and autonomous, ensuring their reliability and predictability becomes paramount. By considering methodologies such as test-driven development and continuous integration, developers can catch defects early in the development cycle, minimizing the risk of costly errors and ensuring that the agents they create are truly fit for purpose.

Developers must also consider best practices for cloud native application development, such as containerization, microservices architecture, and Infrastructure as Code (IaC). By leveraging tools like Docker and Kubernetes, developers can package and deploy their AI agents as lightweight, portable containers, ensuring consistent behavior across different environments and enabling seamless scaling and orchestration.

Furthermore, the integration of open APIs and cloud services like AWS or GCP or Azure into agent-based systems demands a deep understanding of API design principles and best practices for cloud service integration. Developers must carefully consider factors such as API versioning, security, and performance, while also leveraging cloud native patterns and practices to ensure the scalability, reliability, and cost effectiveness of their solutions.

AI Agent Use Cases

There are several areas today in which AI agents are integrated into corporate enterprise portfolios. In this section, we will provide several examples of AI agent applications that we have seen in the enterprise.

Customer Service and Ecommerce

The first and most popular use of AI agents takes the form of customer service chatbots. These bots are implemented on most corporate consumer sites today and range from simple bots with a hard coded script to personalized bots that remember user preferences and conversational history. The move from chat to voice recognition was obvious, and we anticipate that most call centers will implement some version of voice-enabled AI agents in their call centers within the next couple years.

AI has been used in the form of recommender systems in ecommerce since the mid-2000s, and we are now seeing AI agents harness this capability in their ecommerce customer service bots, providing interactive and customized recommendations for ecommerce customers. From agents that help you choose clothing based on your personal style to agents that package industrial purchases based on specific business needs, many companies are already experimenting with these personalized agents.

Decision Support Systems

AI agents implemented as part of decision support systems can be extremely useful at sifting through large volumes of data to identify anomalies and emerging patterns. When fed quality data and provided with explicit and well-understood incentive structures, decision support agents offer a level of analysis that is impossible for human analysts alone.

While AI tools have been used in data analysis for many years, AI agents can provide a level of configurability and interaction to the executive that has up to now been limited to the purview of trained data engineers.

Intelligent Tutoring Systems

Some cutting-edge schools have introduced tutoring agents to their elementary students as a means of personalized education, identifying areas of struggle, and tailoring their approach to ensure that concepts and lessons resonate with clarity and understanding. Through adaptive algorithms and intelligent user interfaces, AI agents in intelligent tutoring systems can create immersive and engaging learning experiences, assisting teachers and providing additional learning resources to students and parents.

Business Process Automation

From data entry and document processing to workflow management, the reasoning capabilities of AI agents allow businesses to move beyond mere task automation and move to more complex business process automation. BPA in the form of rigid rules-based engines and complex expert systems has been difficult to implement in business critical environments. The learning capacity of AI agents allows for these processes to be more flexible and easier to maintain than previous monolithic business rule engines. These systems can provide reliable decision-making capabilities as well as optimize the efficiency and productivity of human staff by automating complex, repetitive, and error-prone tasks.

IT Operations Automation

Observability and anomaly detection alerting systems form the backbone of IT operations. Overworked systems operations support staff are some of the most enthusiastic users of AI agent applications as they relieve the

stress of having to keep all eyes peeled on every change event and anomaly across the enterprise all by themselves. AIOp agents coordinate together to form a holistic picture of enterprise operations help, remediating common events such as memory overloads and system restarts. SecOps agents also work to automatically apply the latest patches and constantly scan for the newest security vulnerabilities.

Robotic Process Automation (RPA)

Robotic Process Automation (RPA) agents mimic and amplify physical human actions and interactions, automating repetitive tasks across various software platforms. With each keystroke and mouse click, the agent replicates the actions of its human counterparts, extracting data, consolidating information, and generating reports with a speed and precision that transcends the limitations of manual processes. Several RPA tools exist on the market today, and the coordination of AI agents to manage and conduct RPA agents promises new levels of business productivity.

Intelligent Automation

AI agents possess the ability to not only automate repetitive tasks but also to engage in complex cognitive processes and decision-making. AI agents superpass the limitations of traditional automation systems as they are capable of analyzing unstructured data, detecting patterns, and making informed decisions that once belonged solely to the domain of human expertise.

From streamlining business processes and elevating customer experiences, to safeguarding digital infrastructures, AI agents are the driving force behind a transformation that promises to reshape the modern enterprise.

Conclusion

In this chapter, we have continued our discussion of Enterprise AI application technical design with a focus on AI agents. We have discussed the different types of agents and the best practices for designing AI applications that consist of the collaboration of several AI agents working in tandem. We concluded this section with an overview of several use cases in which we see AI agents playing valuable roles in the enterprise.

In the next two chapters, we will be discussing the critical elements of testing and test automation in building your Enterprise AI application.