LEARNING GEN AI

- BY SANJANA MCS

AI AGENTS

Models on their own are trained on limited data. They are hard to adapt, even with additional data and resources.

Let's say we ask ChatGPT:

"What is my website's response time?"

It replies "I cannot directly access real-time data or browse the internet."

This highlights a key limitation of general language models

- However, when we build an AI agent, we make it capable of accessing real-time data.
- It can call functions, use external tools, and mimic human behavior. The agent can take actions based on requests, just like humans do.

All of this happens automatically.

Introduction

AI Agents are advanced systems that combine artificial intelligence (AI), real-time data access, and autonomous decision-making to perform tasks traditionally requiring human intervention. Unlike general-purpose large language models (LLMs) like ChatGPT, AI agents are designed to interact dynamically with the world, execute actions, and adapt to real-time inputs. This report explores the limitations of traditional LLMs, the capabilities of AI agents, and their transformative potential across industries.

Limitations of General-Purpose LLMs (e.g., ChatGPT)

While LLMs excel at generating human-like text, they face critical constraints:

- **No Real-Time Data Access:** LLMs rely on pre-trained datasets and cannot access live information (e.g., weather updates, stock prices, or website performance metrics).
- **Passive Interaction:** They cannot execute actions (e.g., booking a flight, sending emails, or analyzing live data).
- **Limited Contextual Adaptability:** Responses are static and cannot learn dynamically from new inputs without retraining.

What Are AI Agents?

AI agents are autonomous systems equipped with:

- **Real-Time Data Integration:** Access to live databases, APIs, and internet resources.
- **Function Calling:** Ability to trigger external tools (e.g., payment gateways, analytics platforms).
- **Human-Like Behavior:** Natural language interaction, emotional intelligence, and contextual awareness.
- **Decision-Making Autonomy:** Execute tasks (e.g., order fulfillment, customer support) without human input.

Key Components of AI Agents

Component	Description
Data Access Layer	Connects to live APIs, databases, and IoT devices for real-time insights.
Action Module	Executes functions (e.g., sending emails, adjusting server capacity).
Learning Engine	Continuously improves via reinforcement learning and user feedback.
Natural Language Interface	Mimics human conversation for seamless user interaction.

WHY AI AGENTS?

AI agents can dramatically improve productivity and efficiency in several ways:

- Automation of complex tasks that would normally require human intervention
- 24/7 availability for monitoring and responding to events
- Ability to process and analyze large amounts of data in real-time
- Integration with existing tools and systems for seamless workflows

These capabilities make AI agents invaluable for businesses and individuals looking to streamline their operations and reduce manual workload. By handling repetitive tasks and providing intelligent assistance, AI agents free up human resources for more strategic and creative work.

Use Cases

Industry	AI Agent Application
Healthcare	Monitor patient vitals, alert doctors, and schedule appointments.
E-commerce	Recommend products, process returns, and optimize pricing in real time.
Finance	Detect fraud, execute trades, and provide personalized investment advice.
Customer Service	Resolve tickets, book appointments, and escalate complex issues.

Challenges

- **Data Privacy:** Securing sensitive information accessed by agents.
- **Ethical Risks:** Bias in autonomous decisions or misuse of authority.
- **Technical Complexity:** Integrating diverse systems (APIs, legacy software).

Different types of frameworks

CrewAI

What it is: A framework for building **collaborative AI agents** that work together like a team to solve complex tasks.

Key Features:

- **Role-Based Agents**: Define agents with specific roles (e.g., "Researcher," "Writer," "Analyst") and assign them tasks.
- **Task Delegation**: Automatically split workflows into subtasks and assign them to specialized agents.
- **Human-in-the-Loop**: Allows human oversight for critical decisions.
- **Integration**: Works with tools like search engines, APIs, and databases.

Example Use Case:

A content creation team with:

- A **Researcher Agent** to gather data from the web.
- A **Writer Agent** to draft the content.
- A **Reviewer Agent** to check for accuracy.

```
from crewai import Agent, Task, Crew

researcher = Agent(role="Researcher", goal="Find reliable data")
writer = Agent(role="Writer", goal="Draft engaging content")
task1 = Task(description="Research AI trends", agent=researcher)
task2 = Task(description="Write a blog post", agent=writer)

crew = Crew(agents=[researcher, writer], tasks=[task1, task2])
result = crew.kickoff()  # Agents collaborate to complete tasks
```

LangChain

What it is: A framework for building **LLM-powered applications** by connecting language models to external tools and data.

Key Features:

- **Chains**: Link multiple LLM calls, tools, or APIs into a sequence.
- **Tools & Agents**: Integrate external tools (e.g., calculators, search engines) and let agents decide when to use them.
- **Memory**: Maintain context across interactions (e.g., chat history).
- **Data Loaders**: Connect to databases, PDFs, or websites for Retrieval-Augmented Generation (RAG).

Example Use Case:

A customer service agent that:

- Uses a **search tool** to pull product FAQs.
- Invokes an **email API** to send follow-ups.
- Retains conversation history for context.

```
from langchain.agents import AgentExecutor, create_tool_calling_agent
from langchain_community.tools import DuckDuckGoSearchRun

tools = [DuckDuckGoSearchRun()]
agent = create_tool_calling_agent(llm, tools, prompt)
agent_executor = AgentExecutor(agent=agent, tools=tools)
agent_executor.invoke({"input": "What's the latest news about AI?"})
```

phidata

What it is: A framework for building **AI Assistants** with long-term memory, knowledge bases, and tool integration.

Key Features:

- **Stateful Assistants**: Maintain memory across sessions (e.g., remember user preferences).
- **Prebuilt Templates**: Ready-to-use templates for PDF assistants, research agents, etc.
- **RAG Support**: Built-in retrieval from documents, websites, or databases.
- **Multimodal**: Supports text, image, and video interactions.

Example Use Case:

A **PDF Assistant** that:

- Ingests and indexes PDFs.
- Answers questions using RAG.
- Saves chat history for future reference.

```
from phi.assistant import Assistant
from phi.tools.duckduckgo import DuckDuckGo

assistant = Assistant(tools=[DuckDuckGo()], show_tool_calls=True)
assistant.print_response("Summarize the latest AI breakthroughs.")
```

Comparison

These frameworks can work together:

1. Use **LangChain** to connect to tools and build chains.
2. Use **CrewAI** to orchestrate multiple LangChain-powered agents.
3. Use **phidata** to add memory/state management to CrewAI agents.

- **CrewAI**: For multi-agent systems (e.g., simulate teams).
- **LangChain**: For single-agent apps needing custom tools/chains.
- **phidata**: For assistants requiring memory or prebuilt templates.

All three frameworks are open-source and designed to work with LLMs like GPT-4, Claude, or Llama 3.

Framework	Focus	Strengths	Use Case
CrewAI	Multi-agent collaboration	Team workflows, role-based task delegation	Complex projects requiring teamwork
LangChain	LLM integration & tooling	Chaining tasks, RAG, flexible tool use	Single-agent apps with external tools
phidata	Stateful AI assistants	Long-term memory, prebuilt templates	Personalized assistants with memory

Conclusion

AI agents represent the next evolution of AI, combining the linguistic prowess of LLMs with real-world functionality. By automating tasks, accessing live data, and mimicking human behavior, they unlock unprecedented efficiency and innovation. Organizations adopting AI agents will gain a competitive edge in responsiveness, scalability, and user satisfaction.