****

**Introduction of AI Agents**

Done by : Sourour Hammoud

**What are LLMs?**

An LLM is a type of AI model that excels at**understanding and generating human language.** It learns by reading huge amounts of text, so it can understand how language works.

Most modern LLMs use something called a Transformer. It is a special AI design that uses Attention, a method that helps the model focus on the important parts of the text.

There are 3 types of transformers:

1. Encoder: transformer takes text as input and output a dense representation of that text.
2. Decoders: transformer focuses on generating new tokens to complete a sequence, one token at a time.
3. Seq2seq: Encoder-Decoder.

The core idea behind a Large Language Model (LLM) is to predict the next token in a sequence based on the previous ones. A "token" is a unit of text, often smaller than a word, used for efficiency. Instead of using full words, LLMs use a vocabulary of sub-word tokens (example: “interest” + “ing” = “interesting”). This allows the model to work with a much smaller and more flexible vocabulary, like LLaMA 2’s 32,000 tokens, rather than hundreds of thousands of full words.

**Types of Decoders:**

* Greedy Decoding: the easiest decoding strategy would be to always take the token with the maximum score.
* Beam Search: It examines several potential sequences to determine which one has the highest overall score, even if some of the individual elements have lower scores.

The most important input is the prompt. The prompt is the first text given to the LLM. If this text is clear and qualified, you help the LLM to provide the correct answer, not an incorrect or general one.

LLMs are trained on large datasets of text, where they learn to predict the next word in a sequence through a self-supervised or masked language modeling objective.

Chat templates help organize conversations between users and language models. They define how messages are arranged and turned into one clear prompt for the model to understand and respond to properly.

**What are AI Agents?**

AI Agents are systems that enable Large Language Models(LLMs) to perform actions by extending their capabilities by giving LLMs access to tools and knowledge.

An Agent is a system that leverages an AI model to interact with its environment in order to achieve a user-defined objective. It combines reasoning, planning, and the execution of actions (often via external tools) to fulfill tasks.

AI agent has two main parts:

1. The brain (AI model)
2. The body (capabilities and tools)

**Level of agency:**

1. Simple processor: The AI only gives a reply. It doesn’t affect anything else.
2. Router: The AI chooses between options, like yes or no (if else only).
3. Tool caller: The AI decides which function to run and with what data.
4. Multi-step Agent: The AI controls how long the task continues.
5. Multi-Agent: The AI triggers another agent to help with a task.

**Components of system AI Agent are:**

1. Environment:The defined space where the AI Agent is operating.
2. **Sensors:** AI Agents use sensors to gather and interpret the information of the environment about the current state of the environment.
3. Actuators: For the current task, the agent determines what action to perform to change the environment.

## **The Thought-Action-Observation Cycle:**

The AI Agent uses a while loop that begins with thinking (Thought) where the LLM part of the agent determines the next step. It then moves on to acting (Act), where the agent performs an action by calling tools with the appropriate arguments. Finally, there is observation (Observe), where the model reflects on the response received from the tool. This loop continues until the agent's objective has been achieved.

Thought: **.** Two techniques:

1. COT: chain of thought. helps the model **reason internally**, especially for logical or mathematical tasks, **without interacting with external tools.**
2. ReAct: Reasoning + Acting.ReAct is a prompting technique that encourages the model to think step-by-step and interleave actions (like using tools) between reasoning steps.

**.** Multiple type: planning, Analysis, Decision Making, Problem Solving, Memory Integration, Self-Reflection, Goal Setting and Prioritization.

Actions:  **.**Types of Agent Actions:

|  |  |
| --- | --- |
| Type of Agent | Description |
| JSON Agent | The Action to take is specified in JSON format. |
| Code Agent | The Agent writes a code block that is interpreted externally. |
| Function-Calling Agent | It is a subcategory of the JSON Agent which has been fine-tuned to generate a new message for each action. |

**.** Type of Actions:

|  |  |
| --- | --- |
| Type of Action | Description |
| Information Gathering | Performing web searches, querying databases, or retrieving documents. |
| Tool Usage | Making API calls, running calculations, and executing code. |
| Environment Interaction | Manipulating digital interfaces or controlling physical devices. |
| Communication | Engaging with users via chat or collaborating with other agents. |

Observation: **.** Type of observation:

|  |  |
| --- | --- |
| Type of Observation | Example |
| System Feedback | Error messages, success notifications, status codes |
| Data Changes | Database updates, file system modifications,  state changes |
| Environmental Data | Sensor readings, system metrics, resource usage |
| Response Analysis | API responses, query results, computation outputs |
| Time-based Events | Deadlines reached, scheduled tasks completed |

* The concept of agents existed before the creation of LLMs (**Large Language Models**). The advantage of building AI Agents with LLMs is their ability to interpret human language and data.
* Outside of AI Agent, the action of LLMs are limited to text only. Inside of AI Agent, LLMs can accomplish tasks using tools that are available in their environment.
* The tools that LLMs can use are determined by the environment they are working in, or the developer of the AI agent can limit the agent’s tool access.
* The difference between memory and knowledge is that the Memory can be short-term in the context of the conversation between the user and the agent but the Knowledge can be long-term, AI Agent retrieve it from other system, service, tools and even other agents.

**The different types of agents**

1. **Simple Reflex Agents:** Perform immediate actions based on predefined rules.
2. **Model-Based Reflex Agents:** Perform actions based on a model of the world and changes to that model.
3. **Goal-Based Agents:** Create plans to achieve specific goals by interpreting the goal and determining actions to reach it.
4. **Utility-Based Agents:** Consider preferences and weigh tradeoffs numerically to determine how to achieve goals.
5. **Learning Agents:** Improve over time by responding to feedback and adjusting actions accordingly.
6. **Hierarchical Agents:** There are multiple agents. One of these agents is the manager who divides the work into smaller tasks and assigns them to the other agents.
7. **Multi-Agent Systems (MAS):** Agents complete tasks independently, either cooperatively or competitively.

**When to use AI Agents**

1. **Open-Ended Problems: if the task is not clear and can’t be hardcoded into a workflow.**
2. **Multi-Step Processes: when the** AI Agent needs to use tools or information over multiple turns instead of single shot retrieval (task is complex).
3. **Improvement Over Time:** tasks where the agent can improve over time by receiving feedback from either its environment or users in order to provide better utility.

**What are AI Agent Frameworks?**

AI agent frameworks are software platforms designed to simplify the creation, deployment, and management of AI agents. These frameworks provide developers with pre-built components, abstractions, and tools that streamline the development of complex AI systems.

Traditional AI Frameworks can help you integrate AI into your apps and make these apps better in the following ways:

-Personalization: AI can analyze user behavior and preferences to provide personalized recommendations, content, and experiences.

-Automation and Efficiency: AI can automate repetitive tasks, streamline workflows, and improve operational efficiency.

- Enhanced User Experience: AI can improve the overall user experience by providing intelligent features such as voice recognition, natural language processing, and predictive text.

The difference between AI frameworks and AI agent frameworks is that AI agent frameworks not only develop AI to understand text, chatbots, and voice recognition, but also create intelligent agents that can interact with users, other agents, and the environment to achieve specific goals. These agents can make decisions and adapt to changing conditions.

some key capabilities enabled by AI Agent Frameworks:

* Agent Collaboration and Coordination: Enable the creation of multiple AI agents to solve complex tasks.
* Task Automation and Management: Provide mechanisms for automating multi-step workflows, task delegation, and dynamic task management among agents.
* Contextual Understanding and Adaptation: Equip agents with the ability to understand context, adapt to changing environments, and make decisions based on real-time information.

There are some things that are common across most AI Agent Frameworks that can help you quickly prototype and iterate namely module components, collaborative tools, and real-time learning.

**Use Modular Components**: AI Agent frameworks like Microsoft Semantic Kernel and Lang Chain give you ready-made parts like AI connectors, memory tools, and prompt templates. These help teams build smart systems quickly without starting from zero.

**Example:** AI travel booking agent with auto function calling using Semantic Kernel

Use:

* AI Connector: connect kernel with Azure open AI model.
* Prompt+ Chat History: to remember the text in the chat and to respond in the same context.
* Plugin+ Function Module: These encapsulate functions that an application can use. like book-flight (date, location) to confirm the booking.
* Auto Function Calling: AI makes decisions automatically based on user text.

**Leverage Collaborative Tools:** Leverage collaboration tools means using AI tools to create agents that work together. Each agent does a different job, like finding data or making decisions. Together, they solve tasks faster and better.

**Example:** Team of AI Agents Collaborating to Analyze Data

Use: 3 agents:

* Retrieval Data Agent: using retrieve\_tool
* Analyze Data Agent: using analyze-tool
* proxy User Agent: using function UserProxyAgent (“user-proxy”, input-func=input). get direct input from the user on the keyboard.

**Learn in Real-Time: AI agents can learn while working.** They look at what happens, learn from mistakes, and do better next time. This helps them improve over time.

The differences between AutoGen, Semantic Kernel and Azure AI Agent Service are:

* **AutoGen:** is built around the core concept of agents, which are autonomous entities that can perceive their environment, make decisions, and take actions to achieve specific goals. Use in case of Automating code generation, data analysis tasks, and building custom agents for planning and research functions.

Some important core concept of AutoGen:

Agent: an agent is a software entity that communicates via messages, maintains its own state, performs action. **Example:** Create your own agent with Chat capabilities.

Use:

* AssistantAgent: type of Ready Agents.
* TextMessage: type of message that exchange between Agents.
* OpenAIChatCompletionClient: connection with ChatGPT.

Multi agents:  AutoGen supports the creation of multiple agents that can work together to achieve complex tasks.

Agent Runtime: A system that runs agents, manages how they talk, live, and stay secure.

There are two type of Agent Runtime:

1. Stand-alone Runtime: Runs all agents in the same app, same language, same machine. Simple setup.
2. Distributed Runtime: Runs agents in different apps or machines, possibly with different languages.

* **Semantic Kernel+ Agent Framework:** Semantic Kernel SK is an open source SDK that helps to build intelligent, AI powered agents and application by combing LLMs with normal code, memory and external function. Kernel is the brain that manages all interactions between LLMs, memory, functions and planning.

Agent Frameworks is part of Semantic Kernel.

* **Azure AI Agent Service:** is a new platform from Microsoft, designed to help developers build and run intelligent agents. It supports flexible use of various open-source LLMs like Llama 3, Mistral, and Cohere. Aimed at enterprise applications, it offers strong security and reliable data storage. The service integrates easily with multi-agent frameworks like AutoGen and Semantic Kernel, and currently supports development in Python and C#. It’s available in public preview.

Core concept of Azure AI Agent:

Agent: The Azure AI Agent Service works with Azure AI Foundry, where an agent operates as a smart microservice that can be used to answer questions (RAG), perform actions, or completely automate workflows. It achieves this by combining the power of generative AI models with tools that allow it to access and interact with real-world data sources.

Thread and messages: a thread is a conversation between an agent and a user. It keeps track of the context and progress, allowing the interaction to continue smoothly without losing information.

**Integrates with other AI frameworks:** Azure AI Agent Service can work with other frameworks like AutoGen and Semantic Kernel. This lets you build parts of your app using those frameworks and use the Agent Service to coordinate everything, or you can build the entire app within the Agent Service itself.

The difference between these frameworks:

|  |  |  |  |
| --- | --- | --- | --- |
| Framework | Focus | Core Concepts | Use Cases |
| AutoGen | Event-driven, distributed agentic applications | Agents, Personas, Functions, Data | Code generation, data analysis tasks |
| Semantic Kernel | Understanding and generating human-like text content | Agents, Modular Components, Collaboration | Natural language understanding, content generation |
| Azure AI Agent Service | Flexible models, enterprise security, Code generation, Tool calling | Modularity, Collaboration, Process Orchestration | Secure, scalable, and flexible AI agent deployment |

**What is the Tool Use Design Pattern?**

The **Tool Use Design Pattern** focuses on giving LLMs the ability to interact with external tools to achieve specific goals. Tools are code that can be executed by an agent to perform actions.

There are many cases to use these tools:

* **Dynamic Information Retrieval**
* **Code Execution and Interpretation**
* **Workflow Automation**
* **Customer Support**
* **Content Generation and Editing**

**Key Elements of the Tool Use Design Pattern:**

1. Function/Tool Schemas: describe each tools, its name, what it does, input needs and what it returns.
2. Function Execution Logic: Decides when and how to use tools based on the user’s message and the context of the conversation.
3. Message Handling System: Manages the flow between the user, the AI, tool requests and tool responses.
4. Tool Integration Framework: Connect the Agent to actual tools.
5. Error Handling and Validation: Catches problems like missing or incorrect input, failed tool calls and ensure responses make sense.
6. State Management: Tracks conversation context, previous tool interactions, and persistent data to ensure consistency across multi-turn interactions.

When using the Tool Use Design Pattern to build trustworthy AI agents, it's important to consider security, especially when generating SQL queries. To avoid risks like SQL injection or accidental data tampering, you should configure databases with read-only access for the AI agent. Running the app in a secure environment and using a separate, user-friendly data warehouse for read-only access further reduces risks. These precautions ensure the agent can safely use tools without compromising data integrity or system security.

**The Agentic Design Principles:**

**Agentic Design Principles** are guidelines used to design AI agents that interact with people in a helpful, respectful, and human-centered way.

Agentic frameworks aren’t always needed with LLMs. For simple tasks, plain code with prompt chains can be enough, giving developers more control and clarity.

The Agentic design principles:

* **Space: Connecting people, not replacing them.** Agents should support human relationships and collaboration, not take over. Agents help connect events, knowledge, and people. Agents bring people closer together. They are not designed to replace or belittle people. **Easily accessible yet occasionally invisible.**
* **Time:**
* Past: The agent reflects on history, not just past actions but also context, to make better decisions now. It uses memory to connect past events to current situations.
* Present: Instead of just sending notifications, the agent gently nudges the user at the right time with helpful cues. It responds to the user’s intent, environment, and context in smart and adaptive ways.
* Future: The agent evolves with the user. It adapts to different devices, behaviors, and preferences, becoming more useful and personalized over time through continued interaction.

**Agentic Time Principles** guide how an AI agent learns from the past, responds smartly in the present, and evolves over time to better serve the user’s needs.

* **Core:** These are the key elements in the core of an agent’s design., Agents are designed to handle uncertainty but must be built with trust and transparency. The user stays in full control, and the agent’s status is always clear.

**The Guidelines to Implement These Principles:**

1. Transparency: Show how AI works and let users give feedback.
2. Control: Let users personalize and manage the system.
3. Consistency: Keep experiences clear, simple, and familiar across devices.

**What is Guardrails?**

Guardrails is a safety check that controls what the AI can see, do or say.

There are three types of Guardrails:

1. Input guardrails: Check and filter user requests before they reach the AI.

* Relevance Classifier: Redirects off-topic requests to the right topic.
* Safety Classifier: Blocks unsafe or manipulative prompts.
* Moderation: Flags hate speech or harassment before AI sees it.
* Rules based Protections

1. Tool-Based Guardrails: Pause or block tool actions until approved by a human.

* Tool Safeguards: Pauses tool actions (like changing salary) for human approval.

1. Output Guardrails: Clean or adjust the AI's response before sending it to the user.

* PII Filter: Removes personal/private info before sending response.
* Output Validation: Ensures response follows company tone and rules.