High-Level Concepts

This is a quick guide to the high-level concepts you'll encounter frequently when building LLM applications.

Large Language Models (LLMs)

LLMs are the fundamental innovation that launched LlamaIndex. They are an artificial intelligence (AI) computer system that can understand, generate, and manipulate natural language, including answering questions based on their training data or data provided to them at query time. You can learn more about using LLMs.

Agentic Applications

When an LLM is used within an application, it is often used to make decisions, take actions, and/or interact with the world. This is the core definition of an **agentic application**.

While the definition of an agentic application is broad, there are several key characteristics that define an agentic application:

- **LLM Augmentation**: The LLM is augmented with tools (i.e. arbitrary callable functions in code), memory, and/or dynamic prompts.
- **Prompt Chaining**: Several LLM calls are used that build on each other, with the output of one LLM call being used as the input to the next.
- **Routing**: The LLM is used to route the application to the next appropriate step or state in the application.
- **Parallelism**: The application can perform multiple steps or actions in parallel.
- **Orchestration**: A hierarchical structure of LLMs is used to orchestrate lower-level actions and LLMs.
- **Reflection**: The LLM is used to reflect and validate outputs of previous steps or LLM calls, which can be used to guide the application to the next appropriate step or state.

In LlamaIndex, you can build agentic applications by using the `Workflow` class to orchestrate a sequence of steps and LLMs. You can learn more about workflows.

Agents

We define an agent as a specific instance of an "agentic application". An agent is a piece of software that semi-autonomously performs tasks by combining LLMs with other tools and memory, orchestrated in a reasoning loop that decides which tool to use next (if any).

What this means in practice, is something like: - An agent receives a user message - The agent uses an LLM to determine the next appropriate action to take using the previous chat history, tools, and the latest user message - The agent may invoke one or more tools to assist in the users request - If tools are used, the agent will then interpret the tool outputs and use them to inform the next action - Once the agent stops taking actions, it returns the final output to the user

You can learn more about agents.

Retrieval Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is a core technique for building data-backed LLM applications with LlamaIndex. It allows LLMs to answer questions about your private data by providing it to the LLM at query time, rather than training the LLM on your data. To avoid sending **all** of your data to the LLM every time, RAG indexes your data and selectively sends only the relevant parts along with your query. You can learn more about RAG.

Use cases

There are endless use cases for data-backed LLM applications but they can be roughly grouped into four categories:

Agents: An agent is an automated decision-maker powered by an LLM that interacts with the world via a set of tools. Agents can take an arbitrary number of steps to complete a given task, dynamically deciding on the best course of action rather than following pre-determined steps. This gives it additional flexibility to tackle more complex tasks.

Workflows: A Workflow in LlamaIndex is a specific event-driven abstraction that allows you to orchestrate a sequence of steps and LLMs calls. Workflows can be used to implement any agentic application, and are a core component of LlamaIndex.

Structured Data Extraction Pydantic extractors allow you to specify a precise data structure to extract from your data and use LLMs to fill in the missing pieces in a type-safe way. This is useful for extracting structured data from unstructured sources like PDFs, websites, and more, and is key to automating workflows.

Query Engines: A query engine is an end-to-end flow that allows you to ask questions over your data. It takes in a natural language query, and returns a response, along with reference context retrieved and passed to the LLM.

Chat Engines: A chat engine is an end-to-end flow for having a conversation with your data (multiple back-and-forth instead of a single question-and-answer).

Tip

- Tell me how to customize things
- Continue learning with our understanding LlamaIndex guide
- Ready to dig deep? Check out the component guides