# LangChain Core Components

*Visual and Practical Guide for LLM Application Development*

## 1. Introduction

LangChain provides a modular framework for building applications with Large Language Models (LLMs). Its core functionality is organized into six components: **Models, Prompts, Chains, Indexes, Memory, and Agents**. These components allow structured reasoning, data retrieval, memory retention, and tool usage for intelligent applications.

## 2. LangChain Components

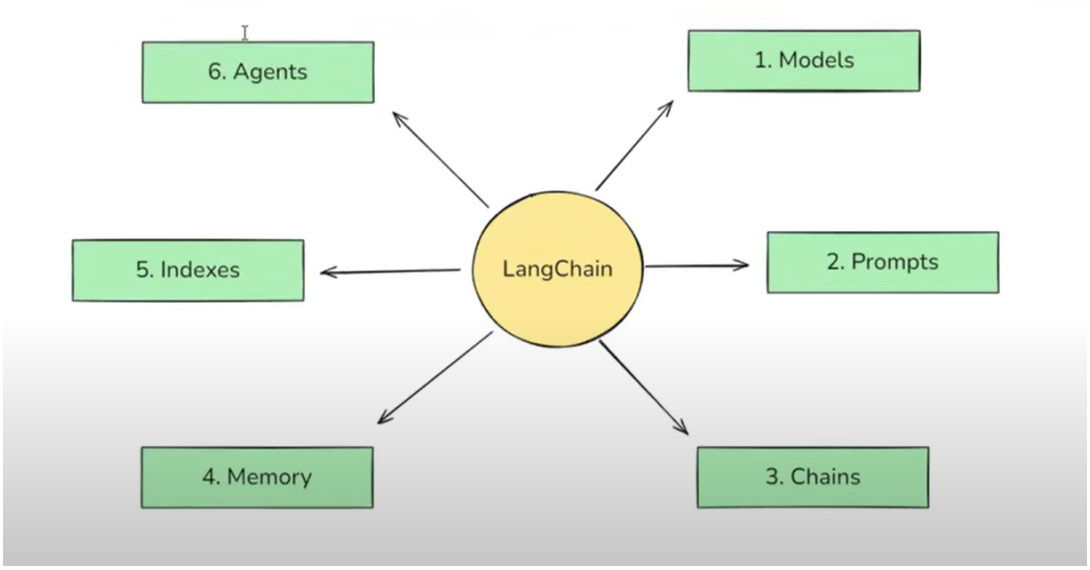


Figure 1: LangChain Components Architecture

## 3. 1. Models

**Models** generate text or embeddings. LangChain supports multiple providers (OpenAI, Anthropic, HuggingFace, Ollama) for LLMs and embedding models.

### 3.1. Example Usage

```
from langchain.llms import OpenAI
llm = OpenAI(temperature=0.5)
response = llm("Explain LangChain simply")
```

## 4. 2. Prompts

**Prompts** structure user input for LLMs. *PromptTemplates* allow dynamic variable substitution, making input reusable and error-free.

### 4.1. Example

Template: "Translate the following text from source_lang to target_lang: text" Usage: "Translate from English to French: Hello World"

## 5. 3. Chains

**Chains** link multiple steps together (prompts, LLM calls, tools) to form multi-step reasoning pipelines.

### 5.1. Example: LLMChain

Prompt: "Summarize: article" Input: "LangChain simplifies LLM applications." Output: "LangChain makes building LLM apps easier."

## 6. 4. Indexes

**Indexes** structure data for efficient retrieval. Used in RAG systems or any LLM requiring external knowledge.

### 6.1. Types of Indexes

- **Vector Index:** Embedding-based similarity search (FAISS, Pinecone).
- **Keyword Index:** Classic inverted index for text search.
- **Hybrid Index:** Combines vector + keyword search for efficiency.

### 6.2. Example Usage

Query: "Benefits of LangChain" → Retriever uses vector index → LLM responds with top documents.

## 7. 5. Memory

**Memory** allows LLMs to remember context or conversation history for multi-turn interactions.

### 7.1. Memory Types

- **BufferMemory:** Temporary session storage.
- **ConversationBufferMemory:** Maintains chat history.
- **VectorStoreMemory:** Embedding-based memory for retrieval.

### 7.2. Example Usage

User: "What is LangChain?" → LLM responds. Next: "How can it help chatbots?" → LLM recalls previous answer using memory.

## 8. 6. Agents

**Agents** enable LLMs to act autonomously by using tools, APIs, or executing code.

### 8.1. Example Tools

- Python REPL
- Google Search API
- WolframAlpha
- Custom internal APIs

### 8.2. Example Usage

Task: "Compute sqrt(256)" → Agent uses Python REPL → Returns '16'.

## Summary

LangChain's **six core components** provide a complete framework to build intelligent, context-aware LLM applications:

- **Models:** Text or embeddings generation.
- **Prompts:** Structured input templates.
- **Chains:** Multi-step pipelines.
- **Indexes:** Data retrieval and organization.
- **Memory:** Context retention.
- **Agents:** Action execution via tools/APIs.