Introduction to LangChain

What is LangChain?

LangChain is an open-source framework for developing applications powered by Large Language Models (LLMs). It provides a comprehensive ecosystem that connects LLMs to external data sources, tools, and memory systems to build production-ready AI applications.

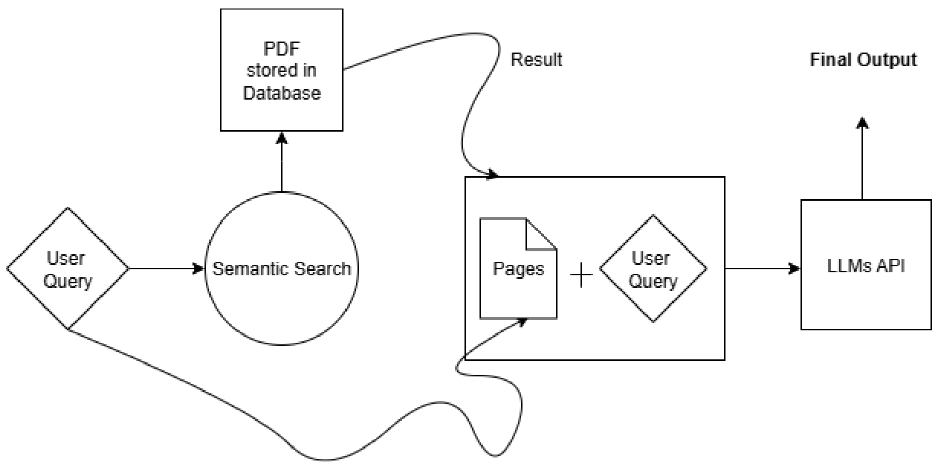
Core Concept

LangChain solves the fundamental challenge of LLM deployment: LLMs are extremely heavy and require specialized infrastructure. Companies like OpenAI, Anthropic, and Google host their LLMs on dedicated servers and provide access through APIs, making them accessible for developers without requiring massive computational resources.

For example,

Scenario: PDF of ML book stored in database
User Query: "Explain page no.5 like I am 5 years old"
System Process:

1. Semantic search identifies relevant content from page 5
2. Context understanding recognizes the need for simple explanation
3. LLM generates age-appropriate explanation
   Output: Simple, clear explanation suitable for a 5-year-old



Semantic Search vs Keyword Search

1. Keyword Search

It gives many irrelevant pages with similar keywords like “assumption”, “linear regression”. It returns pages containing these exact words and is inefficient.

2. Semantic Search

It understands query meaning using vector embeddings. Paragraphs converted to high-dimensional vectors (e.g., 100 dimensions). Query converted to similar vector representation. The system finds closest matching vectors using mathematical similarity.

Result: Returns contextually relevant results, not just keyword matches

Example:

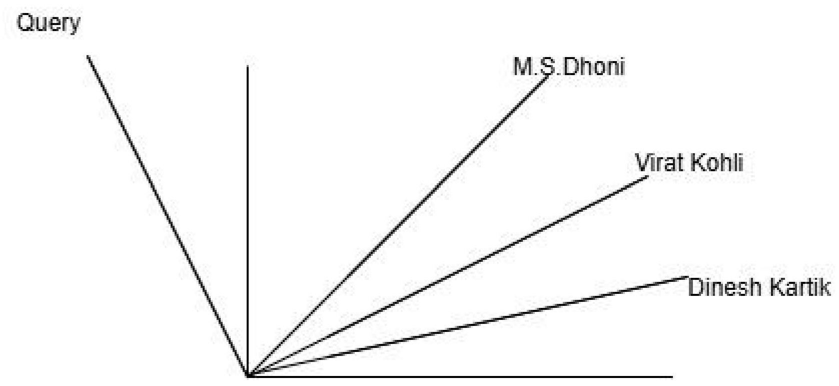
Query "MS Dhoni runs"

Paragraphs about -> M. S. Dhoni, Virat Kohli, Dinesh Kartik

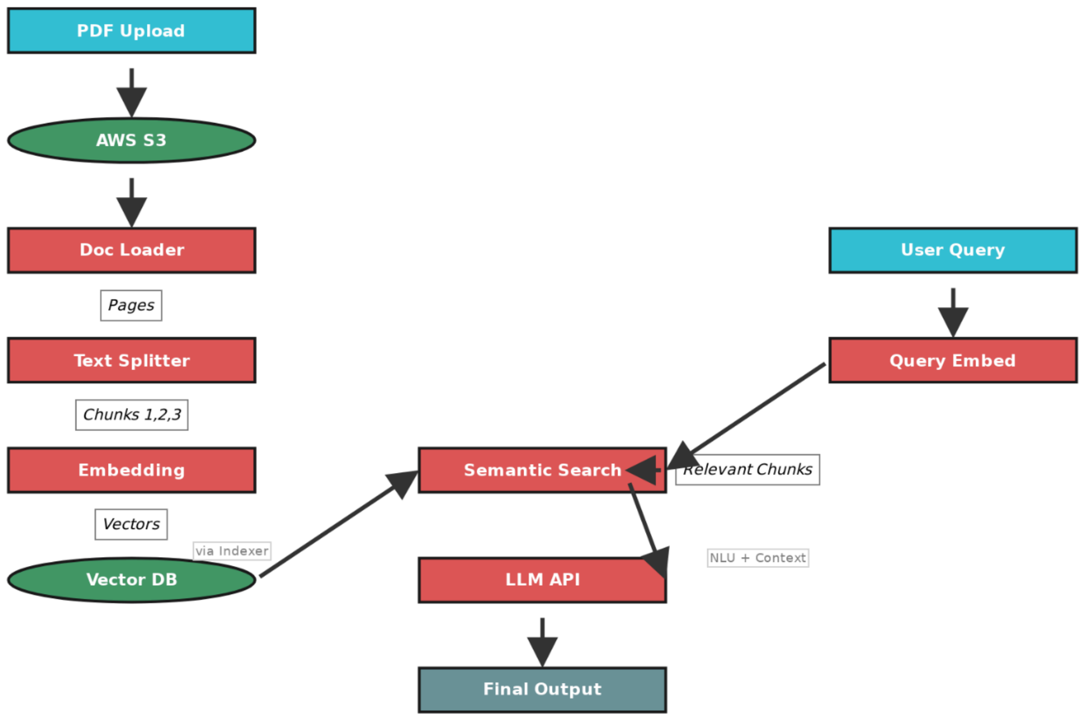
It uses embeddings (vectors) to convert paragraphs into numbers and embedding for the query also.

So, it finds content about cricket statistics, not just pages containing those words

Vectors -> Set of numbers



Detailed Overview



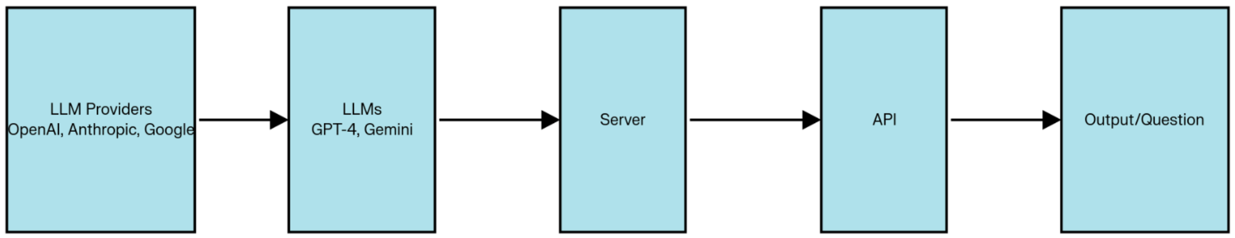
Challenges:

1. Challenge:

Component build <- Query understanding is most important
Relevant Text Generate -> NLU, Context Aware Text Generation
Solutions: LLMs, LLMs API

2. Challenge:

(OpenAI, Anthropic, Google)'s LLMs are extremely heavy, so they store their LLMs on its own server and provide LLMs API.



Key Benefits

1. Concept of Chains → Pipeline

LangChain introduces the concept of "chains" - sequential operations that create processing pipelines for complex tasks.

2. Model-Agnostic Development

- Support for multiple providers (OpenAI, Google, Anthropic)
- Easy switching between different LLM models
- Consistent interface across different AI services

3. Complete Ecosystem

- Document Loaders: Handle various file formats (PDF, TXT, etc.)
- Text Splitters: Break documents into manageable chunks
- Embeddings: Convert text into vector representations
- Vector Stores: Efficient storage and retrieval of embeddings

4. Memory and State Handling

- Store conversation history and context
- Maintain session state across interactions
- Enable contextual understanding in conversations

What can you build using Langchain?

1. Conversational Chatbots

- Scale -> handles large volumes of user interaction (between company -> customers)
- Communication -> Natural Language conversations with context retention
- Integration -> connect with existing business systems

2. AI Knowledge Assistant

- Intelligent information retrieval and explanation.
- Answer complex questions using organizational knowledge.
- Understand and maintain conversation flow

3. AI Agent

Let’s take the example of MakeMyTrip
Suppose, an older man doesn't know how to use this platform -> AI agent will help to book their tickets. And guide users through multi-step processes.

4. Workflow Automation

- Level -> Company wide process automation
- Integration -> Connect multiple systems and databases
- Efficiency -> Reduce manual tasks and improve productivity

5. Summarization/Research Papers

- Content: Research papers, books, long documents
- Output: Concise, accurate summaries
- The company’s large data is not allowed to upload directly on LLMs. In that case, you can create your own chatbot tool -> to process huge data & answers easily to related questions.

Frameworks & their Features

Feature	LlamaIndex	LangChain	Haystack
Best for	Data preparation and indexing for LLMs, fast retrieval from large datasets.	Building complex applications, AI agents, and orchestrating multiple tools.	Production-ready search systems, scalable Q&A, and hybrid retrieval.
Primary strength	Optimizing data for LLM input and fast querying.	Flexibility and composing complex LLM workflows.	Pre-built, production-grade search and QA pipelines.
Ideal use case	Integrating LLMs with your own data efficiently.	Chatbots, conversational agents, and multi-turn interactions.	Building scalable and accurate search products.
Flexibility vs. Opinion	More focused on the data layer, less on end-to-end logic.	More flexible, but can require more setup for search-centric tasks.	More opinionated, providing turnkey solutions for search.
Production readiness	Growing in popularity for RAG and data-focused tasks.	Widely adopted for building diverse applications.	Considered a stable and production-ready framework, especially for search.