# Langchain

## Architecture:

### <https://python.langchain.com/v0.2/docs/concepts/#langchain-expression-language-lcel>

A screenshot of a computer

Description automatically generated

### Langchain core :

The interfaces for core components like LLMs, vectorstores, retrievers. No third party integrations are defined here

### Partner Packages

 long tail of integrations are in langchain-community, we split popular integrations into their own packages (e.g. langchain-openai, langchain-anthropic

### Langchain package:

Chains , agents, retrieval strategies - general

### Langgraph:

langgraph is an extension of langchain aimed at building robust and stateful multi-actor applications with LLMs by modeling steps as edges and nodes in a graph.

### Langserve:

Deploy langchains as REST apis

### Langsmith :

A developer platform that lets you debug, test, evaluate, and monitor LLM applications.

## **LangChain Expression Language**

1. Declarative way to chain langchain components
2. LLM response is streamed directly (with incremental chunks) to output parser , no delay
3. Async support (in Langserve)
4. Optimized parallel execution ( chains in parallel)
5. Retries and fallback : make reliable at scale
6. Access intermediate results: debug chain with intermediate results
7. Input and output schemas: pydantic and jsonschema – validate input and output
8. Seamless langsmith tracing -> as chain get complex , trace intermediate steps
9. Seamless deployment

### Runnable protocol:

To ease custom chains , “Runnable” protocol was created . Most Langchain components implement this. A screenshot of a computer

Description automatically generated A screenshot of a computer program

Description automatically generated

## Components:

Chat models , LLMs , Messages , Prompt templates , Example selectors, Output parsers , chat history , documents , document loaders , text splitters , embedding models , vector stores , retrievers , tools , Agents , callbacks

### Chat models:

Language models that use a sequence of messages as inputs and return chat messages as outputs (as opposed to using plain text. Chat models support the assignment of distinct roles to conversation messages, helping to distinguish messages from the AI, users, and instructions such as system message. Although the underlying models are messages in, message out, the LangChain wrappers also allow these models to take a string as input. This means you can easily use chat models in place of LLMs. Message is wrapped as humanMessage and passed to the model.

Multimodality: images , audio , files , videos as inputs .

Check how to guide : <https://python.langchain.com/v0.2/docs/how_to/#chat-models>

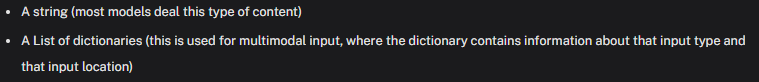
### LLMS:

Language models take string input and return string output. Although underlying model in s in , s out , Langchain wrapper wraps them into messages and give the same interface as chat model

### Messages:

Some language models take a list of messages as inputs and returns messages. There are a few different types of messages. All messages have a **role**, **content**, and **response\_metadata** property.

Role : who is saying the message . ex: human , system , AI etc

Content : 

HumanMessage represents message from user

AIMessage represents message from model

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

### Prompt Templates:

Prompt templates translate user input and params into instructions for LLMs which will guide response, helping it to understand context and generate appropriate response

Prompt template takes input as dict , where each key represents variable in the prompt template to fill.

Types of prompt templates:

A screenshot of a computer program

Description automatically generated A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated A screen shot of a computer

Description automatically generated

### Example Selectors:

Provide example prompts to achieve better results. Example selectors are classes responsible for selecting and formatting examples into Prompts

### Output Parser:

Takes outputs from models and transform it into suitable formats for down stream tasks. Used to provide structured response.

Lang chain supports some output parsers A screenshot of a computer

Description automatically generated

Formats:

Json , XML , CSV , OutputFixing -> wrapes another output parser , if the output has error , this will pass the error message and the bad output to an LLM and ask to fix it , RetryWithError -> similar to above , it shall pass input additional to them , pydantic -> data formating , YAML , pandas df , Enum , Date time , Structured .

### Chat History:

LLM has conversational interface , conversational system should be able to access window of some past messages. ChatHistory wraps past messages to a list of messages and pass them for future interactions.

### Documents:

A black background with white text

Description automatically generated

### Document loaders:

These load document objects , Langchain has integrations with various data sources . A screen shot of a computer

Description automatically generated

### Text Splitters

Loaded documents often need to be transformed to suite our application in a better way. Split long document in smaller chunks that fit into model’s context window with small overlap between them to contain the context. Langchain has built in transformers which make it easy to split , combine , filter and manipulate.

A screenshot of a computer

Description automatically generated

### Embedding models:

A black screen with white text

Description automatically generated A diagram of a model

Description automatically generated A black background with white text

Description automatically generated

2 methods internally . embedding documents and embeding queries

### Vector Stores

Most common ways to store and search over unstructured data is to embed it and store the resulting embedding vectors, at query time embed the query and retrieve embedding vectors similar to embedding query. Performs vector search. Most vector stores can also store meta data about embedded vectors and supports filtering on metadata giving more control over returned docs.

A black screen with white text

Description automatically generated

### Retrievers:

Retrievers returns doc given unstructured query. Not same a vector store as it need not store documents but retrieves them from vector stores, Wikipedia , amazon kendra etc

Accepts string query as input and returns list of document outputs

### Tools:

Interfaces that an agent , chain or LLm use to interact with world. A screenshot of a computer error

Description automatically generated A black background with white text

Description automatically generated

### Tool Kits:

A screenshot of a computer

Description automatically generated

### Agents:

Language models can’t take actions on their own, the big use case is creating agents. Agents use LLMs as reasoning engine to determine which actions to take and what are the inputs to such action. The result of the action is fed back into the agent and it determines whether more actions are needed or no.

Langraph is specially aimed at creating highly controllable and customizable agents. Legacy agent concept (AgentExecutor) will be deprecated

### Call backs :

LC provides callback systems that allow you to hook into the various stages of your LLM . Used for logging , monitoring , streaming etc

You can subscribe to these events using callback arguments available throughout the API .

A screenshot of a computer

Description automatically generated A screenshot of a computer screen

Description automatically generated

Request time callbacks are passed at the time of request and are inherited by all children

Constructor callbacks are passed as args in the constructor call and are not inherited and scoped only to the object defined.

## Techniques:

### Streaming :

LLMs generate outputs iteratively which helps showing intermediate results.

#### Callbacks:

A screenshot of a computer

Description automatically generated A screenshot of a computer program

Description automatically generated

Stream is simpler than callbacks for all the above reasons

We need not provide additonal config for stream which are needed for callbacks

But stream returns only on etype of value , which is ok for single LLM calls but for more complex chains you would want the intermediate values example when we need to return sources along with final text. You can do this with .assign() or .astream\_events()

A screenshot of a computer

Description automatically generated A screenshot of a computer program

Description automatically generated

### Structired Output

We need to constraint the LLM’s output to a specific format or structure, If not it would give us wide variety of formats.

A black screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

#### Raw Prompting:

Ask nicely for the format needed. In addition to the query give instructions describing the kind of output , then parse using the output parser to convert raw model message or string output into something easily manipulated

Pros:

1. No special model features , only reasoning to understand passed schema
2. Prompt for any format you like . json , xml , yaml

Cons :

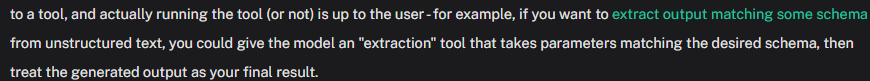
1. Non deterministic , consistency in output not expected
2. Depends on the individual model and trained info

#### JSONmode:

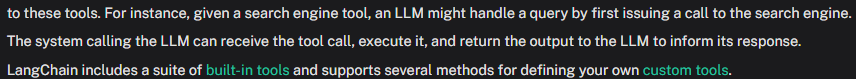
Some models support JSON mode with enabled via config. Which add something along the line “you must always return json” into the prompt A screenshot of a computer program

Description automatically generated

#### Function / Tool Calling:

Tool calling allows model to respond to a given prompt by generating output matching the user defined schema. Model comes up with arguments to the tool and actually running the tool is upto the user. 

Tool calling is convenient and removes guess work.

 A screenshot of a computer

Description automatically generated

### Retrieval:

A screenshot of a computer screen

Description automatically generated

A black and white screen with text

Description automatically generated with medium confidence A black and white screen with white text

Description automatically generated

### Text Splitting:

A screenshot of a computer

Description automatically generated A screenshot of a computer program

Description automatically generated A screenshot of a computer

Description automatically generated