Delevoping LLM Application with langchain

Langchain Ecosystem

→ open source framework for connecting
- Large Language models
- Data sources
- Other functionality under a unified syntax.

→ Allows for scalability
→ contains modular components.

core components of langchain

Open source models — closed source models.

1. prompts
2. parsers & Indexers
3. chains and agents.

Hugging face

→ open-source repository of models datasets and tools.
→ with hugging face API

langchain unifies them both into a consistent, modular workflow

↙ open source model

↘ closed source model.

Real world usage:-

* natural language conversations with documents
* Automate tasks
* Data Analysis.

Prompting strategies for chatbot

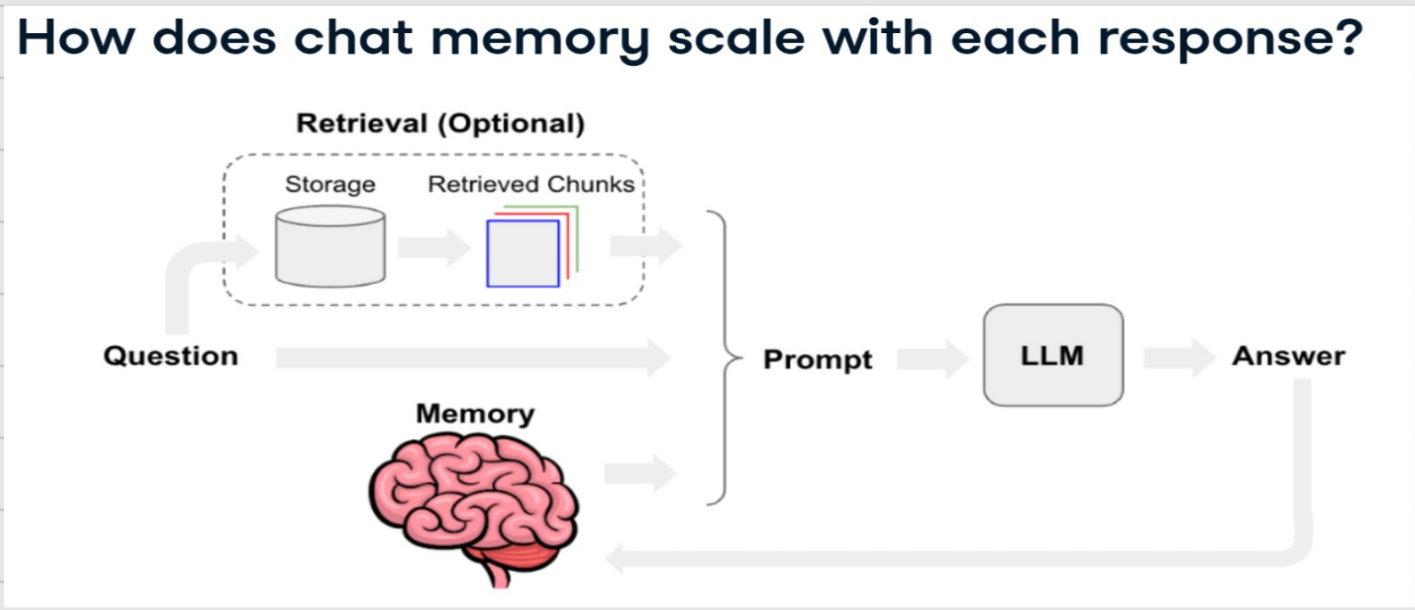
→ fine-tuned models for more domain specific use cases

prompt template: Recipes for generating prompts.
→ flexible and modular
→ can contain: instructions, examples and additional context.

different component are combined together in Langchain using chain

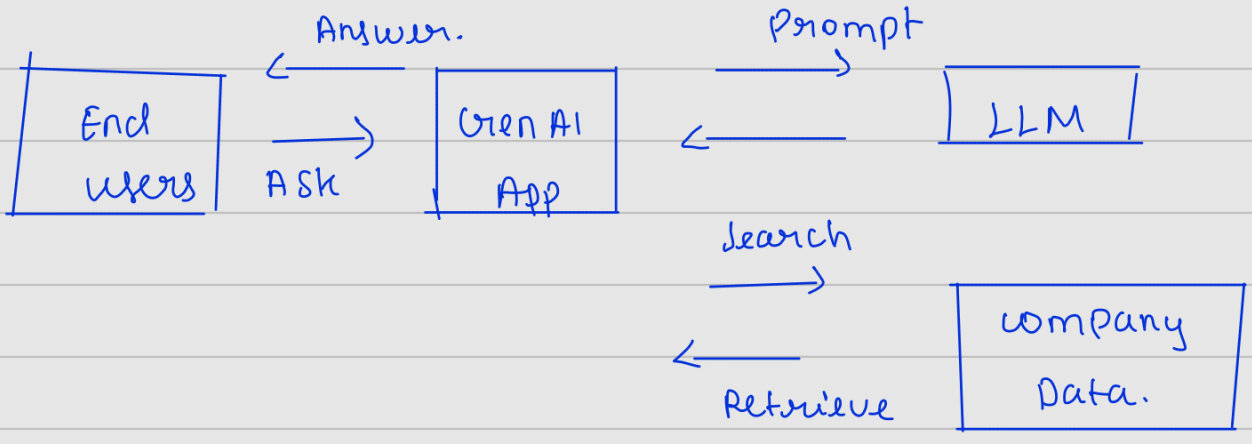
Managing chat model memory

→ follow up questions
→ Response iteration and expansion
→ personalization



Integrating document loaders

Retrieval Augmented Generation (RAG), Pre-trained model do not have access to private sources.



RAG development steps are threefold

Document Loader

↓

Splitting → Storage and Retrieval →

Splitting external data for retrieval

If the model shows the sign of losing context and Misunderstanding information, when answering from external sources ⟶ increases the chunk overlap.

CharacterTextSplitter ⟶ Splits based on the separator

RAG Storage and retrival using vector databases

consideration (vector database)

- ⟶ open source vs closed source (licenses)
- ⟶ cloud vs on-premesises
- ⟶ Lightweight vs powerful for large filestores

Langchain Expression language (LCEL), chains & Agent

Why LCEL?

- ⟶ part of langchain toolkit
- ⟶ Easier chaining of prompt, model and retrieval components.

- ⟶ Effective for production Environment
- ⟶ These chains have built-in support for batch processing, Streaming and asynchronous execution.

- ⟶ Integrate nicely with langsmith & langserve.

Runnables in LCEL

Runnables ⟶ functions / actions executed during expression.

Examples:

* RunnablePassthrough
  (pass inputs to the model)

* RunnableLambda, (transform inputs)
* RunnableMap, (processing inputs in parallel)

Implementing functional langchain chains

Chain categories

- **Generation** chains
  - Example: `ChatOpenAI`, `ChatAnthropic`
- **Retrieval** chains
  - Example: `WikipediaRetriever`, `Chroma`
- **Preprocessing** chains
  - Example: `StrOutputParser`

Sequential chains

↳ output from one call becomes the input for another call.

RunnablePassthrough in chains

↳ passing values between chains.

Agents ——→ High level thought processes.
↳ use language models to decide which actions to take

Tools ——→ functions used by the agent to interact with the system (utilities, chains, more agents)

primary agent components

1. user input in the form of a prompt
2. Definion for handling the intermediate steps
3. Tools and model behaviour definion
4. output parser.

utilizing tools in langchain

custom tools to enhances agent capabilities.

custom tools : user-defined functions for agents to complete tasks.

Examples:

① custom tools ——→ Define single-function tools.
② StructuredTool ——→ Allows for more complex tool definions.
③ format_tool_to_openai_function
↳ format custom tools to use with openai language models.

Tools required for OpenAI model

- input_name
- output_name
- function_name
- tool_name
- description.

Troubleshooting methods for optimization

callbacks → function / methods called during execution of a program.

Used for
- checking output
- optimizing
- Troubleshooting.

callbacks for AI Application

1. Data preprocessing
2. Model inference stage.
3. Error handling & logging
4. Resource management
5. user interaction management.

Evaluating model output in langchain

Benefits of AI evaluation

→ checks accuracy.
→ identifies strengths and weakness
→ Re-align model output with human intent.

langchain evaluation tool

* Built-in evaluation tools
  (common criteria, including: relevance and correctness)

* custom evaluation criteria
  (criteria for a particular use case)

* using QAEvalchain
  (configuring the model to choose for best optimization)

langsmith → troubleshooting and evaluating application

langServe → deploying applications.

langGraph → Multi-agent knowledge graph.