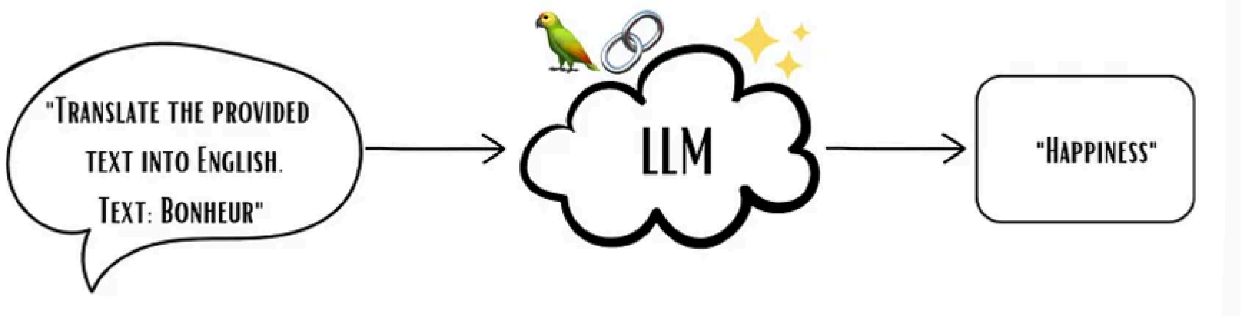
Selected Topics in AI: Week 8
LangChain

What is LangChain?

- **LangChain** is a framework for developing applications powered by large language models (LLMs).
- LangChain isn't itself a large language model (LLM), but rather a tool that helps developers build applications that use LLMs. Think of it as a Lego set for LLMs.
- It includes integrations with a wide range of systems and tools. LangChain allows chaining of various modular components, such as a PromptTemplate and LLM, to create a sequence or Chain that takes an input, processes it, and generates a response.
- This makes it a versatile tool that can be used in a variety of different contexts and applications.
- LangChain incorporates seven modules: **Prompts, Models, Memory, Indexes, Chains, Agents, and Callbacks.**

Prompts



- A prompt refers to the statement or question provided to the LLM to request information. Typically, it is a short text snippet or a few sentences that convey the user's intent or query.
- The Prompts module is responsible for the creation and management of prompts.
- LangChain simplifies the creation of prompts by providing a PromptTemplate. This template can be thought of as a format or pattern for the prompts, with placeholders that can be filled with specific details or examples. This approach allows reusing of

prompts, which becomes especially important as the prompt length increases.

- There are two types of prompt templates:
    - Text prompt templates
    - Chat prompt templates.

Models

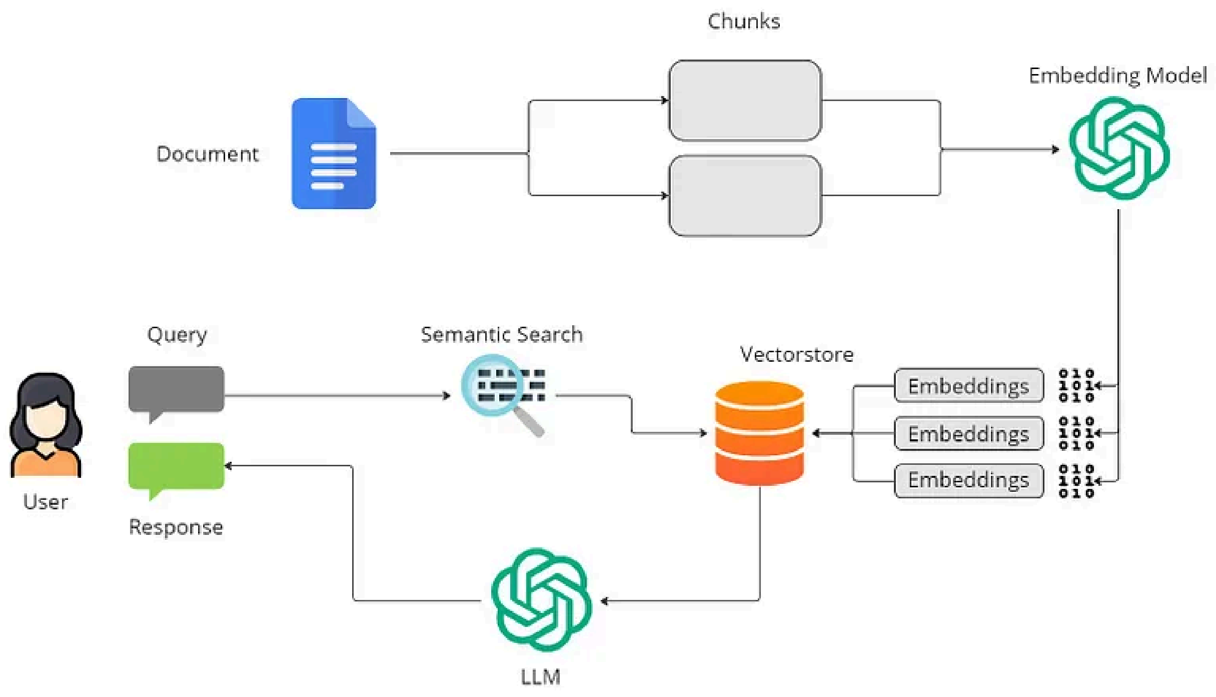
The Models module is a core component of LangChain. It consists of two types of models:

- Language Models
    - Language Models are particularly well-suited for **text generation tasks.** There are two variations of language models:
        - **LLMs**
        - Chat Models.
    - **LLMs** take a text string as input and output a text string as well.
- Text Embedding Models.

Memory

- The Memory module in LangChain is designed to retain a concept of the **state** throughout a user's interactions with a language model.
- **Statelessness** means that the language model treats each incoming query **independently.**
- However, in certain applications like chatbots, it is crucial to remember previous interactions at both short-term and long-term levels.
- Passing the entire conversation as a context to the language model is not an optimal solution, as the number of tokens can quickly increase and potentially exceed the allowed limit.
- The **Memory** module **facilitates the storage of conversation history,** addressing these challenges.

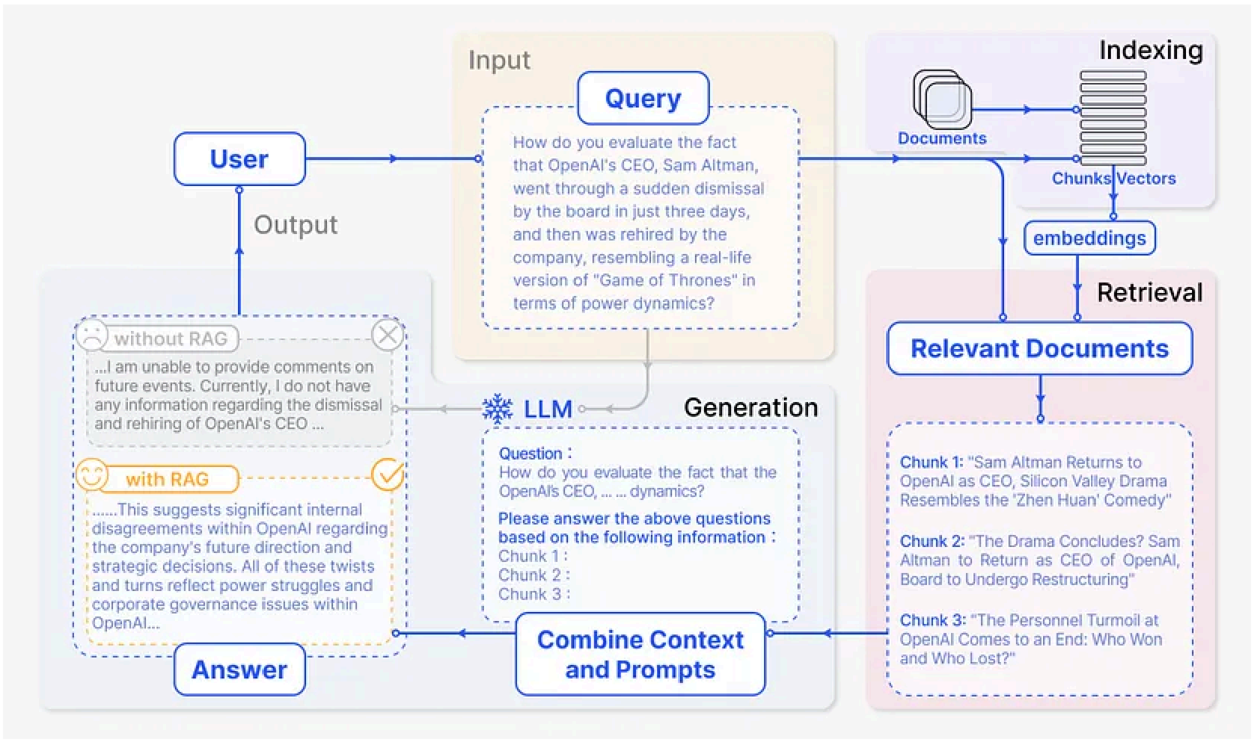
RAG (Retrieval Augmented Generation)



- Large Language Models (LLMs) demonstrate significant capabilities but sometimes generate incorrect but believable responses when they lack information, and this is known as “hallucination.” It means they confidently provide information that may sound accurate but could be incorrect due to outdated knowledge.

- **Retrieval-Augmented Generation or RAG** framework solves this problem by integrating an information retrieval system into the LLM pipeline. Instead of relying on pre-trained knowledge, RAG allows the model to **dynamically fetch information from external knowledge sources** when generating responses.
- This dynamic retrieval mechanism ensures that the information provided by the LLM is not only contextually relevant but also accurate and up-to-date.
- It is a more efficient way to provide additional or domain-specific information using an external database, rather than repeatedly retraining or fine-tuning the model on updated data.

How does RAG work?


Input
Query
How do you evaluate the fact that OpenAI's CEO, Sam Altman, went through a sudden dismissal by the board in just three days, and then was rehired by the company, resembling a real-life version of "Game of Thrones" in terms of power dynamics?
Indexing
Documents
Chunks
Vectors
embeddings
User
Output
Retrieval
Relevant Documents
without RAG
...I am unable to provide comments on future events. Currently, I do not have any information regarding the dismissal and rehiring of OpenAI's CEO ...
LLM
Generation
with RAG
......This suggests significant internal disagreements within OpenAI regarding the company's future direction and strategic decisions. All of these twists and turns reflect power struggles and corporate governance issues within OpenAI...
Question :
How do you evaluate the fact that the OpenAI's CEO, ... dynamics?
Please answer the above questions based on the following information :
Chunk 1 :
Chunk 2 :
Chunk 3 :
Chunk 1: "Sam Altman Returns to OpenAI as CEO, Silicon Valley Drama Resembles the 'Zhen Huan' Comedy"
Chunk 2: "The Drama Concludes? Sam Altman to Return as CEO of OpenAI, Board to Undergo Restructuring"
Chunk 3: "The Personnel Turmoil at OpenAI Comes to an End: Who Won and Who Lost?"
Answer
Combine Context and Prompts

- **Indexing**
  - **The indexing process** is a crucial first step in data preparation for language models. Original data is cleaned, converted into standardized plain text, and **segmented** into smaller chunks for efficient processing. These chunks are transformed into **vector representations** through an **embedding model,** facilitating similarity comparisons during retrieval. **The final index stores these text chunks and their vector embeddings,** enabling efficient and scalable search capabilities.
- **Retrieval**
  - When a user asks a question, the system uses the encoding model from the indexing phase to transcode it. Next, it calculates **similarity scores between the query vector and vectorized chunks within the indexed corpus**. The system **prioritizes** and **retrieves the top K chunks** showing the highest similarity, using them as an **expanded contextual basis** to address the user's request.

- **Generation**
    - The user’s question and chosen documents are combined into a clear prompt for a large language model. Then the model crafts a response, adapting its approach based on task-specific criteria.